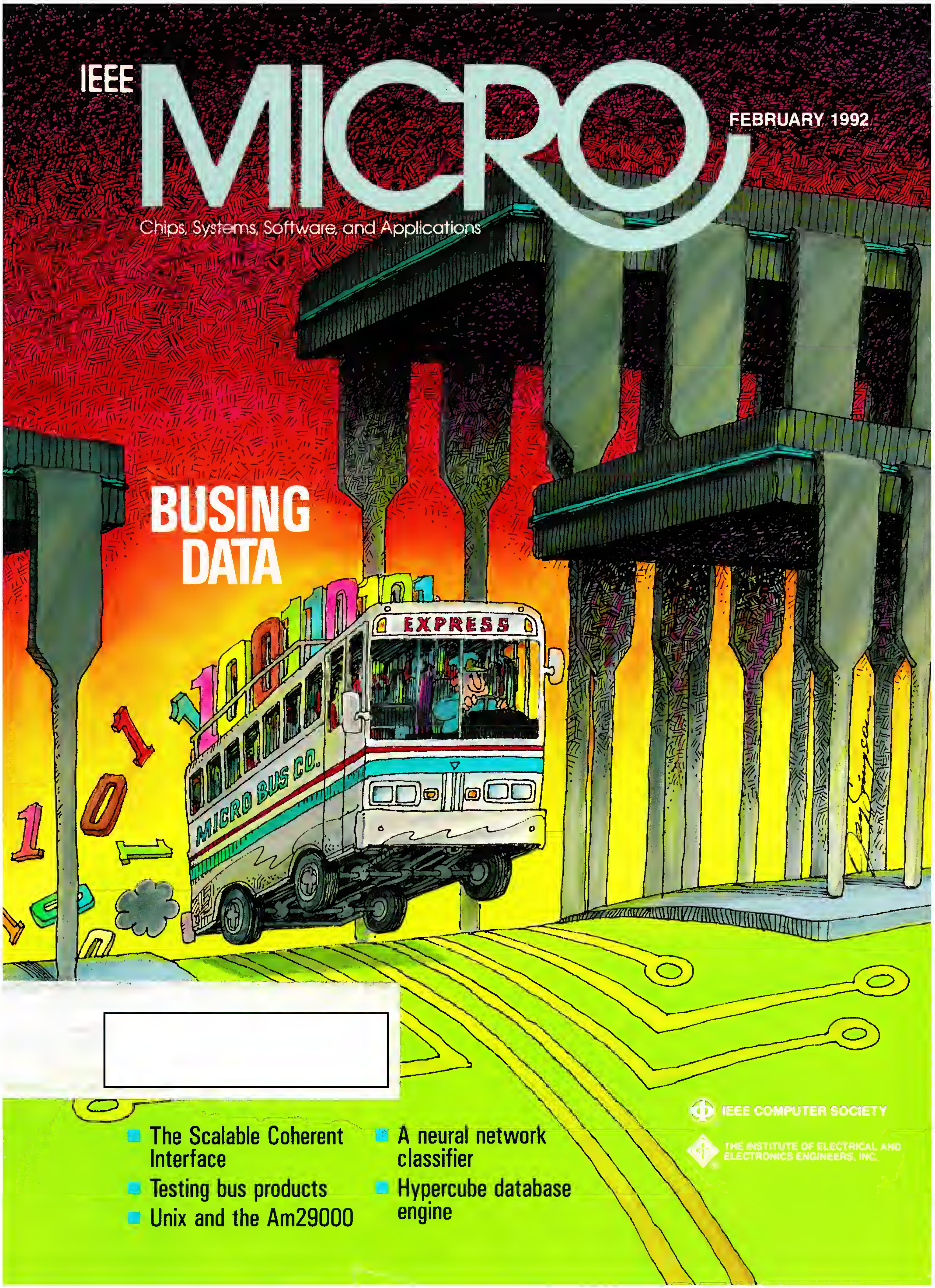
IEEE MICRO

FEBRUARY 1992

Chips, Systems, Software, and Applications



BUSING DATA

- The Scalable Coherent Interface
- Testing bus products
- Unix and the Am29000
- A neural network classifier
- Hypercube database engine

IEEE COMPUTER SOCIETY
THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

IEEE MICRO

Published by the IEEE Computer Society

February 1992

F E A T U R E S

Cover design: Jay Simpson, Design and Direction

Circulation: *IEEE Micro* (ISSN 0272-1732) is published bimonthly by the IEEE Computer Society, PO Box 3014, Los Alamitos, CA 90720-1264; IEEE Computer Society Headquarters, 1730 Massachusetts Ave., NW, Washington, DC 20036-1903; IEEE Headquarters, 345 East 47th St., New York, NY 10017. Annual subscription: $23 in addition to IEEE Computer Society or any other IEEE society member dues; $39 for members of other technical organizations. This journal is also available in microfiche form.
Postmaster: Send address changes and undelivered copies to *IEEE Micro,* PO Box 3014, Los Alamitos, CA 90720-1264. Second-class postage is paid at New York, NY, and at additional mailing offices.

DEPARTMENTS

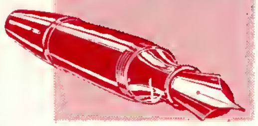
IEEE Computer Society
PO Box 3014, Los Alamitos, CA 90720-1264
(714) 821–8380

* Submit six copies of all articles and special-issue proposals to Dante Del Corso, Dipartimento di Elettronica, Politecnico di Torino, C.so Duca degli Abruzzi, 24, 10129 Torino, Italy; phone +39 11 564 4044; Compmail: d.delcorso; Internet: delcorso@polito.it; Bitnet: delcorso@itopoli

or

** Ashis Khan, Mips Computer Systems, Inc., 950 DeGuigne Drive, Sunnyvale, CA 94086; Internet: ashis@mips.com.

From the Editor-in-Chief



One year later

NOW THAT I'VE BEEN *IEEE Micro*'s EIC for one year, it is time to check the balance sheets; in my introduction of February 1991 I expressed some of my "good intentions." The last two words in the first paragraph in that introduction (I'm sure you keep old issues, so you will know what they were) were surely true.

In this year some things have been accomplished, while others still must be done. I am sure that *IEEE Micro* has been a good channel to bring a lot of useful information to you readers. I myself learned a lot of things. First of all I learned how important the work of the managing editor and staff is to the magazine. I understand now that a magazine could survive without an EIC but not without the staff.

I also learned that Europeans can directly communicate with people in the US only in a narrow time window (and that everybody in the US makes use of an answering machine!). Some big organization should start a project to unify the time on a world basis. (It would be quite easy: You modify the sun into a ring-shaped variable star with the earth in the center.... Sometimes I think that such a project could be simpler than trying to make a magazine exactly as we think it should be.)

Some members of the editorial board completed their terms—we must thank them for their contributions—and others joined the board. Our editorial board is growing with capable people from academia and industry, while it keeps the international balance between the US, Europe, and the Far East.

A magazine results from the combined efforts of authors, editors, staff, and readers. The first three groups of people depend on the last—you, the readers—to continue to support the magazine by providing proposals, comments, and suggestions. As in the past, we promise to always consider these comments.

What changed in the magazine during 1991? Costs were reduced (credit to our staff), allowing us to bring you the most pages possible in spite of inflation and general economy problems.

We have also devoted efforts to reducing the amount of time spent in reviewing manuscripts. The shorter review cycle serves the needs of authors (their work reaches a wide audience in a shorter time, or at least they know their fate sooner) and readers (they receive updated information quickly). The theme issues in 1991 brought coordinated sets of articles on some of the hottest areas in microelectronics and microsystems. This will continue in 1992 and 1993. (See p. 41 for clues.)

The current issue is a "general" one, so you find an assortment of articles that address various aspects of high-performance computing. The first, "The Scalable Coherent Interface and Related Standards Projects" by Dave Gustavson, provides an insight into an IEEE project (P1596-SCI). SCI proposes new solutions for the communication bottleneck intrinsic in bus-based multiprocessor architectures. Instead of wider and faster buses (which also means more expensive and power-hungry buses), we can switch to nonbused interconnection schemes and still keep the configurability we generally find in bused systems. SCI is actually a set of projects addressing the many aspects of an interconnection structure, from the lowest level (the physical

0272-1732/92/0200-0002$03.00 © 1992 IEEE

communication channel) to higher protocol layers dealing with cache coherency. It ensures tight coordination with other standards projects.

Next Daniel Mann's "Unix and the Am29000 Microprocessor" article shows how the hardware features of a RISC processor can be exploited to optimize its performance for a given operating system.

With the third article we switch to completely different architectures. "Hardware Requirements for Neural Network Pattern Classifiers: A Case Study and Implementation" by Boser et al. describes in detail an ASINC, or application-specific integrated neural circuit. This is so new an acronym that it does not yet appear in Carl Warren's glossary (see Micro Standards beginning on p. 69).

Efficient and affordable handwritten character recognition can open the way to a variety of new industrial applications. I am convinced that we will see such features within the sensor itself in a few years (the same way keyboards now have embedded controllers that send ASCII characters directly to the host microprocessor). There is a good chance that such intelligent sensors will use the neural approach, so take a close look at this article to get an idea of what will be on the market in the near future.

A common denominator for these three articles is that each of them achieves high performance, thanks to some specific silicon basis (protocol chips, RISC, neural hardware). The next article, "Experimentation with Hypercube Database Engines" by Frieder et al., describes efficient algorithms for handling distributed shared data in a specific architecture.

Even the best hardware and the best software can provide good results only if they can work correctly. In a modular system, this means each module must comply with the specification of the common interface (the bus). The last article in this issue, "Conformance Testing of VMEbus and Multibus II Products" by Adams et al., describes a technique used to verify compliance of boards to a given bus specification.

Dante Del Corso

In the mailbag

(LK: liked; DLK: disliked LTS: like to see)

In this mailbag the total number of readers who did not like split articles amounts to nine. Their comments are not individually reported here; I think they already received their answer in December. Thank you all for the advice.—D.D.C.

February 1991

LK: Hardware design; LTS: network systems.—S.T., Moscow

April 1991

LK: Optical architecture: excellent article, though I think there were some errors in certain figures and not enough clarity in parts of the text. (Please be more precise, so we can inform the author and ask for corrections if required.—D.D.C.) LTS: RISC architectures.—J.C.A.A., Tlalnepantla, Mexico

LK: Software and hardware computer science and its electronics; LTS: ... U.P.S. units for computers.—M.M.M., Alexandria, Egypt

LK: Letters, Micro News, Micro View, and other chapters; LTS: a chapter [identifying] computer idioms.—M.G., Isfahan, Iran (An acronym dictionary appears in this issue.—D.D.C.)

June 1991

LK: Richard Stern, Book Review—T.S., Tromso, Norway

LK: On The Edge; it has focused very well on the problem. Moreover, author used simple language to describe it ... very important ... to teach something. I'd like to see more articles like this.—C.G., Verona, Italy (These compliments are for Carl Warren and James Gafford, who were responsible for this column.—D.D.C.)

LK: The whole issue; DLK: "Light at the end of the chunnel." It has nothing to do with the hot chips [theme]. (You are correct: Departments are not necessarily linked with special themes.—D.D.C.) LTS: Hot Chips III—T.P., Warsaw (You will get it in the April issue.—D.D.C.)

LK: The overview of the BTRON/286 specs.—E.D., Aalen, Germany

LK: Enjoyed reading [about] iWarp and Datawave. How much does the iWarp and ITT Datawave chip cost? (The task of the designer—who writes the articles—is to minimize the manufacturing cost, but they usually have little control on final pricing. It is better to ask a commercial representative.—D.D.C.) DLK: the guest editors' introduction of what is superscalar and pipelining; LTS: How superscalar and pipelining come into play. Do all chips have one or could they have both? This was not clear.—A.R.F., San Ramon, CA (Almost all microprocessors announced in the last five years are pipelined; only a few recent ones are also superscalar.—M.D.H./D.A.W., guest editors.)

DLK: Rate monotonic scheduling, too little tutorial information.—P.F., Adliswil, Switzerland

LK: iWarp: A 100 MOPS....—S.R.E, Saudi Arabia

August 1991

LK: Micro Law; I never miss it.—T.D.L., Cupertino, CA

LTS: More practical software reports for science and engineering, Unix.—F.M.R., Munich

Micro Law

Engineers can be disqualified, too.

Richard H. Stern

Oblon, Spivak, McClelland, Maier & Neustadt, P.C.

1755 Jefferson Davis Highway

Suite 400

Arlington, VA 22202

During the heyday of the mergers-and-acquisitions frenzy, the disqualification ploy became a common litigation tactic. The first thing a party would do in a lawsuit would be to file a motion that the other side's lawyer should be disqualified from representing his or her client, because at some time or another counsel had performed some kind of legal representation of the adversary party who made the motion.

The rationale for filing such motions, and for the court when it went along with the move, was to prevent the appearance of impropriety. Never mind that the earlier legal representation had nothing to do with the present case. The public's great confidence in the legal profession might be impaired if it even appeared that clients might not be able to rely on the confidentiality of their relationship with their attorneys. How could clients feel safe hiring lawyers and telling them their secrets if the very same secrets would later be used against them to help an adversary?

After a while, the courts became as cynical about these motions as those who filed them and recognized them as another ploy to put the other party at a tactical disadvantage. They became skeptical about arguments based on the body blow to our social order wrought by the appearance of impropriety and began to limit disqualification to situations where a party's adversary would actually get the advantage of relevant business secrets disclosed in confidence in an earlier case. Such things as knowing the structure of the client's hierarchy of values, the client's aversion to or enthusiasm for risk, how the client thinks about things in general, or the client's negotiating strategies, all went out the window as bases for disqualification.

The rule now seems to be pretty close to this: The lawyer must really know (or be likely to know) where the relevant bodies are buried. For example, the court may disqualify counsel if the lawyer

- wrote the patent application on the invention that the lawyer now wants to assert (on behalf of an adversary of the inventor) is invalid;
- is going to be a witness, because he or she wrote the patent application that supposedly was used to defraud the patent office;
- represented the patent owner in another case involving a related patent, where counsel learned about the weaknesses of the client's claim on this technology; or
- represented one party to a license in working out and drafting the agreement, where parties are now in a dispute over what the agreement was intended to do (what it means), and the lawyer wants to represent the other party in the dispute (or maybe now the lawyer plans to claim that the original license agreement is legally invalid).

While things are pretty well sorted out on that front, a new frontier is opening up, which may be of greater concern to electrical engineers and computer science professionals. It now appears that they, too, can be disqualified.

The Balde case. In a decision handed down last summer, *Wang Laboratories, Inc.* v. *Toshiba Corp.*,[1] a federal court in Alexandria, Virginia, (see box) disqualified a computer consultant from representing NEC (a codefendant of Toshiba) because he had previously given Wang a preliminary opinion that its patent was invalid. In November 1990, Wang's attorney had telephoned John Balde, a computer consultant. He asked

0272-1732/92/0200-0004$03.00 © 1992 IEEE

Balde if he was familiar with SIMM (single in-line memory module) technology, and Balde said he was. The parties differ over what happened next. According to Wang, the lawyer retained Balde and agreed to pay him for his time. According to Balde, he was not retained, and he told the lawyer he would have to determine whether Wang's SIMM patents were valid before he would enter any agreement with the company.

Wang's lawyer then sent Balde a letter (dated November 14) transmitting various documents: the SIMM patents, some prior art publications, some materials concerning patent infringement, and a long memorandum written by Wang's attorney about the history of the prosecution of the patents before the patent office. He asked Balde to review the material, "so that we can discuss how best to explain the advantages to a computer designer of using" SIMM, and he suggested they meet after Balde reviewed the material.

The next day—watch out for this one, readers—Wang's lawyer sent a second letter. Unlike the first one, it was prominently labeled, "Confidential Attorney-Work Product." This letter (dated November 15) contained an outline of potential legal defenses against Wang's suit, as Wang's lawyer perceived them, and asked Balde to provide his opinion on various issues. It also said that this material "will assist your review of the material included in my November 14, 1990, letter."

Several telephone conferences followed. The lawyer later said that in them he disclosed Wang's confidential information to Balde, and that he made the confidentiality clear to Balde. Balde did not deny this, but said that he made no use of the material in the November 15 letter. He considered the letter premature because it asked him to give opinions on specific litigation issues. But he did not want to give any opinions on the issues until he had made up his mind about the validity of the SIMM patents. If they were not valid, he did not want to become involved with Wang. Balde felt that Wang's lawyer was jumping the gun on enlisting Balde in Wang's camp before he felt ready to sign up. Nevertheless, as the judge pointed out in his opinion, Balde did not write back to Wang's lawyer saying any of this.

Balde studied the matter and concluded that the SIMM patents were invalid. On December 10 he telephoned Wang's lawyer, told him his conclusion, and said that he therefore did not want to act as a consultant for Wang in the case. The lawyer asked Balde for a report, which Balde sent two days later. In his cover letter for the report, Balde wrote, "As you know, I have read ... the Work-Product information on the two Wang SIMM patents," and he thanked Wang for offering to pay his $1,500 invoice for time spent on the task. Subsequently, NEC retained Balde as its technical expert in the case. Wang then moved to disqualify Balde.

Alexandria's "rocket docket"

Last summer, the US District Court for the Eastern District of Virginia awarded Wang $3.3 million in damages against Toshiba and NEC for patent infringement. The case is of special interest because it comes from the "rocket docket" of Alexandria, Virginia. This court is becoming a forum of choice for patent infringement litigation and other complicated cases that usually drag on for many years, because of its firm policy that all cases *must* be tried within six months after joining issue. Presumably, one can expect the same kind of rulings about experts in other cases brought in that court.

Court's decision. The court found little precedent about disqualification of experts, but held that it had an inherent power to disqualify them in appropriate cases. This power exists to help the court fulfill its judicial duty to protect the integrity of the adversary legal process and promote public confidence in the fairness and integrity of the legal process.

The court found two issues: 1) Was it reasonable for Wang to think it had a confidential relationship with Balde? 2) Did Wang disclose confidential information to Balde? If and only if both answers are affirmative, Balde should be disqualified. It also recognized that lawyers might seek to disable potentially troublesome experts merely by retaining them without using them, and said that it would not countenance that ploy. (Here, Balde had been NEC's consultant in the past.) On the other hand, consultants who do not want to be bound by a duty of confidentiality should make it clear that they do not want to be retained until they make up their minds, and in the meantime they should not accept confidential disclosures.

In this case, the November 15 letter carried the day for Wang. The letter was captioned "confidential attorney-work product," and an examination of the enclosures confirms that description. The court said, "No experienced litigator would freely disclose these materials to opposing counsel." Balde's silence in the face of the November 15 letter established that Wang's lawyer was reasonable in thinking that a confidential relationship existed, and in fact that letter transmitted confidential information to Balde.

Advice. Neither Wang nor Balde had "acted inappropriately," the court found. Even so, it decided to offer some free advice for others "to avoid a repetition of these unfortunate circumstances." First, lawyers should make it clear and confirm in writing that a confidential relationship will be created. Preferably, this should be specifically explained in a letter, along with confirmation of payment terms and conditions. These elements were missing in

this case. Nevertheless, the court disqualified Balde.

Next, consultants should take care to avoid creating confusion about their position. Doubts about wanting to be retained should be unequivocally expressed. Here, the court said, "given his stated [read that as 'alleged'] concerns" about patent invalidity, Balde needed no more than the identity of the parties and the patent numbers. (The last is a bit of judicial overkill. Since the patents are public documents, their text is no secret. It would be an inconvenience to make Balde get his own copies.) He should have declined to accept anything more, the court said.

Counsel should ask the consultant whether his past employment creates any problem. Since NEC apparently did not make this inquiry of Balde before retaining him for the case against Wang, the court felt that NEC had only itself to thank for its problem of having no expert on the eve of trial. (Presumably, Wang, too, should have asked Balde whether his past work for NEC gave him any of NEC's secrets, which could make it improper for him to be Wang's consultant.)

Also, NEC should have promptly advised Wang and discussed the matter "thoroughly in an effort to resolve the dispute before it is raised in court." (To which I say, lots of luck. Why would Wang agree to roll over on anything? Excess of gentlemanliness? When was the last time you heard of noblesse oblige being a significant factor in how litigators behave?)

To be sure, the court said, experts "are not advocates; they are sources of information and opinions in technical [matters] ... Yet, when experts are retained in connection with litigation, they must operate within the constraints of, and consistent with, the adversary process."

I am not going to express any opinion on whether Wang's lawyer sandbagged, hoodwinked, or otherwise mistreated Balde and NEC, nor on whether Balde got a deserved comeuppance. The court apparently felt that none of that happened, which is, of course, quite good enough for me. But the lawyer certainly outmaneuvered the EE in this case.

Protecting yourself. So where does all of this leave other EEs? What do you do to "avoid a repetition of these unfortunate circumstances"? Or, exercising 20-20 hindsight, what should Balde have done? What would the *CYA Manual for Electronic Engineer and Computer Science Would-be Expert Witnesses* have prescribed, if anything? Remember, Balde was not trying to scare Wang away. He didn't know whether NEC would want to retain him in this case, and he had to pay the rent every month. He also did not know, presumably, how he felt about the SIMM patents and Wang's case.

... consultants ... could be "Balde-ized."

Ideally, Balde would have had his own lawyer, whom he would have consulted immediately after the November 14 telephone call. He should have turned the November 14 and 15 letters over to a lawyer, unread and preferably unopened. Balde's lawyer would have responded for him or advised him how to respond. He would have matched Wang's lawyer's efforts with the same thing in reverse, something like the following:

F, U, & D, Counselors-at-Law
November 16, 1990

Dear Sir:

My client, Double-E Balde, has turned over to me your letters of November 14 and 15 and enclosures, which he has not read. My client wishes you to be advised that he is not yet fully prepared, at this time, to enter into a relationship of confidentiality with regard to the subject matter of Wang v. *Toshiba and NEC. In all fairness to Wang, my client feels that he should first make his own preliminary evaluation of materials already of public record, or otherwise not secret or confidential, in order to satisfy himself that he can appropriately consult with, or act as an expert for, Wang in this matter. I am sure you will appreciate that it would not be in Wang's or his interests to have him prematurely enter into a confidential relationship with Wang, in the event that it later appears he would be obliged to testify as to opinions inconsistent with Wang's position in this matter.*

We will get back to you as soon as Mr. Balde has finished the foregoing preliminary evaluation. In the interim, I will retain the documents, unless you wish me to return them to you at this time.

Sincerely yours,
F, U, & D

Of course, you may feel it is not feasible to run a consulting business this way. What then? You consultants who don't want to run your business through lawyers still must do something, or you could be "Balde-ized." At the very least, you should send a letter, such as the following, that negatives a confidential client relationship like the one set up for Balde by the lawyer's letter.

Bit Bucket Consulting Services, Ltd.
November 16, 1990

Dear Counselor:

Thank you for your letters of November 14 and 15, 1990, regarding the pending patent infringement suit between Wang and Toshiba/NEC regarding SIMM technology. I am responding to both letters at the same time, because your second letter arrived before I could

continued on p. 72

R&D in Japan

David K. Kahaner

US Office of Naval Research, Far East

kahaner@xroads.cc.u-tokyo.ac.jp

These two reports and two announcements reflect some of the ongoing research in this country.

Forecast for 2010

Japan's Economic Planning Agency enlisted the assistance of a group of 10 experts to assess the country's position in technology and its direction for the next 20 years. After listing 101 technological items, the study group developed a questionnaire concerning them, combined the responses, and wrote a lengthy report and summary (in Japanese). The summary, released in July 1991, proved fascinating in the detailed views of the group members but produced some problems for readers. For example, the study may not have much statistical validity, given that one or at most two experts assessed the individual items. In addition, parts of the textual material were awkwardly phrased.

But to me, the most interesting portions of the report appear in its tables. Table 1 lists selected items from the report's tables.

Micromachines

Under Japan's National Research and Development Program (popularly known as the Large-Scale Project), industry, government, and academic circles cooperate on research and development of innovative, advanced, large-scale industrial technologies deemed important and urgent for the national economy. Since the program began in 1955, 29 projects have been launched, eight of which continue today. MITI (Japan's Ministry of International Trade and Industry) proposed an R&D program called Micromachine Technology to begin in fiscal year 1991.

The New Energy and Industrial Technology Development Organization (NEDO) under the authority of the Agency of Industrial Science and Technology will conduct the Micromachine Technology project. NEDO plans to establish the technologies necessary for the realization of micromachines.

Elemental technologies. The first step of the program (covering the first four to five years) emphasizes the establishment of basic technologies of elemental components for micromachines and targets four major R&D items. The first major item in this phase will establish technologies for material processing and control of dynamic mechanisms by developing the elements of a micromachine. The elements are actuators (thermal-, electrical-, or magnetic-induced deformation, electrostatic, hydraulic) and sensors (electromagnetic, chemical).

A second item calls for the development of technologies that transform energy from external sources, micro internal batteries, or micro power generators. A third will investigate micromachine communication with the exterior world and conduct theoretical research and associated software development of remote control and coordinated distributed control.

A final major item concerns the measurement of accuracy and movement of the micromachines for evaluation of the results of the R&D program.

R&D description. Though micromachines are small, they are complex systems with advanced functions. Some of the areas that need to be researched include the basic theories that underpin miniaturization methods, structural analysis, materials, and component technologies of microscopic processing and assembly. Other areas include the techniques for producing microscopic sensors and control circuits, and the system technology required to perform microscopic motion

0272-1732/92/0200-0007$03.00

Table 1. Identified technologies and application year.

Technology		Year
Information electronics		
	Biosensor	2000
	Superparallel computer	2010
	Terabit optoelectronic file	2010
	Superintelligent chip	2010
	Terabit optocommunication device	2010
	Automatic translation systems	2020
	Virtual reality system	2020
	Self-replicating database system	2020
	Neurocomputer	2030
	Terabit memory	2030
	Self-replicating chip	2050
New materials		
	Ceramics gas turbine	2000
	Magnetic materials	2010
	Hydrogen occlusion alloy	2010
	Optical IC	2010
	Superlattice devices	2010
	Nonlinear optoelectronics	2020
	Superconductors	2030
Automation		
	Micromachines	2010
	Concurrent engineering	2010
	Intelligent CAD	2020
Communication		
	TV conference system	1994
	TV telephone	1994
	Optoelectronic LAN	1995
	Broadband ISDN switches	1995
	HDTV	1995

and operation. Table 2 on the next page lists some of the main topics envisioned for micromachine technology.

The technology. Microscopic machines or instruments with advanced functions can perform minute tasks or work in extremely narrow spaces. Their small size allows them to be applied to a wide variety of areas, including medicine, biotechnology, and industry.

Recently, silicon micromachining technology, in which micrometer mechanical structures are formed on silicon wafers, opened up this field. The technology emerged from etching, deposition, and other lithographic techniques for microprocessing silicon developed in the 1970s and enabled the production of cantilevers, diaphragms, and other simple mechanical components. These products are now used widely as pressure sensors or are beginning to be commercialized as acceleration and flow sensors. Moreover, recent advances in semiconductor microprocessing and ultraprecision processing technology allow the creation of mechanical parts far smaller than anything previously developed.

Researchers only recently began to investigate micromachine technology, however, and need to overcome many technological barriers before such machines can be developed. Some barriers are those related to friction, durability, strength, materials, and power sources and supplies. Researchers also must develop a number of other technologies, including ways to design, process, assemble, and control micromachines.

Micromachines today. We can approach micromachine development in two ways. One approach uses technology in the field of mechanical engineering, which makes existing mechanisms even smaller; the other uses Micro Electro Mechanical Systems (MEMS) technology, which uses IC production technology. Many researchers have proposed or built prototypes of various microactuators and microstructures that could serve as the component technologies for micromachines.

In the United States, schools like MIT, Stanford, and the universities of Wisconsin-Madison, Michigan, and California, Berkeley continue to research surface micromachining technology and LIGA (Lithografie, Galvanofonnung, Abfonnung) process technology in their silicon and LIGA process research centers.

MIT researchers produced a 100-μm-diameter polysilicon micromotor rotating at 15,000 rpm and analyzed its movement. Researchers created the motor using the same process used for IC production. Wisconsin-Madison researchers produced 3D structures containing gears with 55-μm inside diameters. Researchers at Berkeley produced a 120-μm-diameter electrostatic motor and verified that it does rotate. They can also measure friction coefficients, one of the most difficult problems in micromachine technology, using a micro electrostatic linear actuator.

The US National Science Foundation supports these efforts. In 1988 NSF distributed its micromachine research budget among eight universities and in 1989 provided funding to 11 universities.

Europe also supports several micromachining facilities, including Germany's FraunhoferInstitut, Techische

Table 2. Main topics of micromachine research.

Technologies	Description
Microscopic mechanical device	R&D on structures, materials, machining techniques, integration techniques, and power supplies for microscopic mechanisms and functional components required for micromachines Development of technology to enable the production of various mechanical devices
Microscopic sensors, control circuits, and other techniques for miniaturized electronic devices	R&D in the technologies needed to produce extremely miniaturized electronic devices such as microscopic sensors and control circuits used in micromachines
Control and operation	R&D in motion control and operation technologies for microscopic mechanisms
Measurement and evaluation	Basic research into measurement methods, evaluation methods, and microscopic measurement technology as they relate to various component devices
Support	Basic research into support technologies including lubrication techniques for microscopic parts, theoretical simulation, and CAD/CAM
System integration	R&D project extending 10 years (FY1991-FY2000) and costing approximately 25 billion yen

Universitat Berlin, Kernforschungszentrum Karlsruhe Institut fur Mikrostrukturetechnik, and the Netherlands university of Twente. Research concentrates mostly on sensors. The Fraunhofer Institut fur Mikrostrukturetechnik produced prototypes of a vibration sensor with 32 cantilever mechanical resonators and a 1.5-mm × 1.25-mm cantilever thermal bimorphic microactuator. These facilities receive subsidies both from the European Economic Community and from their respective countries.

Japan pursues numerous creative studies related to micromachine technology. For example, the University of Tokyo developed prototypes of a micro Stirling engine with high thermal efficiency as a microactuator, Tohoku University produced a microvalve using silicon. NTT Applied Electronics Laboratories produced a 500-nm × 500-nm, active integrated optical microencoder with 0.01-μm resolution.

Research at many universities, national research institutes, and private companies continues to produce prototypes. These include electrostatic linear actuators, micropressure sensors, micro IS-FET (ion-sensitive field-effect transistor) sensors, micromanipulators using piezoelectric-impact drive systems, micro active catheters using shape memory alloy (SMA), and more.

These technologies, which mainly use semiconductor production techniques, represent only a small part of micromachine production technology. Research in this field is still in its infancy, and many problems remain to be solved. Some of the hurdles to be overcome include the development of production and process technologies geared specifically toward micromachines and solving questions related to friction, durability, strength, materials, power supplies, and control.

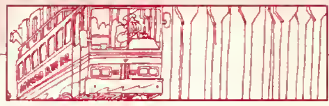
Application of results. Since micromachine technology will have a variety of applications, the program will focus on common component technologies. Once developed these areas probably will result eventually in applications like the following.

Industrial micromachines. Industry faces the need to boost reliability and reduce maintenance costs for ever more-advanced and complex mechanical systems and equipment (power plants and airplane engines are two good examples). A tremendous need exists for technology that makes it possible to perform inspections and repairs in extremely tight spaces without having to dismantle the entire system or equipment in question, such as plant pipe systems and airplane engines.

Industrial micromachines will enable inspection and repair without requiring that plant equipment be dismantled. This capability will make it possible to perform early inspection and repair to minimize the extent of damage. We can therefore expect significant improvements in capacity utilization and maintenance costs for electric power plants and other facilities.

Medical micromachines. Today's medical procedures do not sufficiently alleviate the pain experienced during diagnosis and treatment. In addition, as the population ages, we will see a strong demand for advanced medical equipment that lessens the physical and mental stress inflicted on patients.

continued on p. 87



The Scalable Coherent Interface and Related Standards Projects

The Scalable Coherent Interface (IEEE P1596) provides bus services by transmitting packets on a collection of point-to-point unidirectional links. Its protocols support cache coherence in a distributed shared-memory multiprocessor model, with message passing, I/O, and LAN communication taking place over fiber optic or wire links. Several ongoing SCI-related projects apply the SCI technology to new areas or extend it to more difficult problems.

David B. Gustavson

Stanford Linear Accelerator Center

The Scalable Coherent Interface (SCI) was developed by a number of bus designers and system architects who had come to understand the fundamental limits to bus technology during their work on Fastbus (IEEE 960) and Futurebus+ (IEEE 896.x). These modern buses push bus signaling technology to its limits and provide various architectural features that support the use of multiple processors.

These bus limits are rapidly becoming a serious problem as the demand for computing power continues to grow. The economic reality is that we can only meet this demand by using a large number of fast microprocessors. Buses, however, are inherently a bottleneck (only one transfer at a time), and their signaling speed is limited by the imperfect transmission lines that result from bus-style connections.

Therefore, buses can't support a large number of processors, especially not fast ones. While we can extend their useful life a bit by cleverness and brute force, the potential gains are relatively small and the costs become very high. For example, doubling the width of a bus does not double its speed because there are fixed overheads associated with arbitration and addressing. Lengthening block transfers to reduce the effect of these overheads is of little use once the blocks exceed the size of cache lines.

We can increase signaling speeds by shortening the bus, but that makes it less useful. Reducing the signal voltage helps, but eventually this solution experiences noise problems. Using multiple buses to achieve more than one transfer at a time results in a complex (expensive) bus-bridge mechanism to maintain cache consistency (coherence) in shared-memory systems that use bus-snooping technology.

Paul Sweazey (Futurebus cache-coherence task group coordinator) started the Superbus Study Group in November of 1987 to see if there were potential solutions to these problems. In July 1988 the outline of the solutions had become clear, and the P1596 SCI working group replaced the study group. The work was essentially completed by January 1991, when specification draft D1.00 went out for ballot. Since then, it has been undergoing minor improvements, polishing, and debugging of the specification C code. (Most of the SCI specification is executable, to reduce ambiguity, simplify testing, and enable accurate simulation.)

The resulting draft D2.00[1] recirculated to the balloting body in December. Since the draft passed with 92 percent affirmative, and all but one objection has been resolved, (we refused to change the C code to Pascal), final approval by the IEEE Standards Board seems probable in March 1992, unless new objections arise.

0272-1732/92/0200-0010$03.00

SCI goals

The SCI design goals include

- *Scalability*, so that the same mechanisms can be used in high-volume, single-processor (or few-processor) systems such as one might find in desktop machines, as well as in large, highly parallel multiprocessors (next-generation supercomputers);
- *Coherence*, to support the efficient use of cache memories in the most general and easiest-to-use multiprocessor model, distributed shared memory; and
- An *interface*, a standardized open communication architecture that allows products from multiple vendors to be incorporated into one system and interoperate smoothly.

Scalability keeps costs down, not only through increased volume of production but through the simplicity of having to learn only one new paradigm—one that will work over several generations of machines. (See box.)

SCI applications

SCI uses point-to-point links to achieve very high speed communication. For the highest performance over short distances (typically within a cabinet), 16-bit-wide links run at 1,000 Mbytes/s. For I/O applications within a room, serial coaxial cable links run at 1,000 Mbps. For I/O over campus distances of a few kilometers, optical fibers can carry the same serial bit stream. See Figure A.

SCI's scalable architecture allows the same protocols to cover the range from internal communication within a multiprocessing supercomputer to local area network applications. LAN communications look like moving data from one address to another in memory, a very simple software model. However, wide-area networks need hierarchical addressing instead of SCI's flat 64-bit address model, so the usual software protocol translations are necessary.

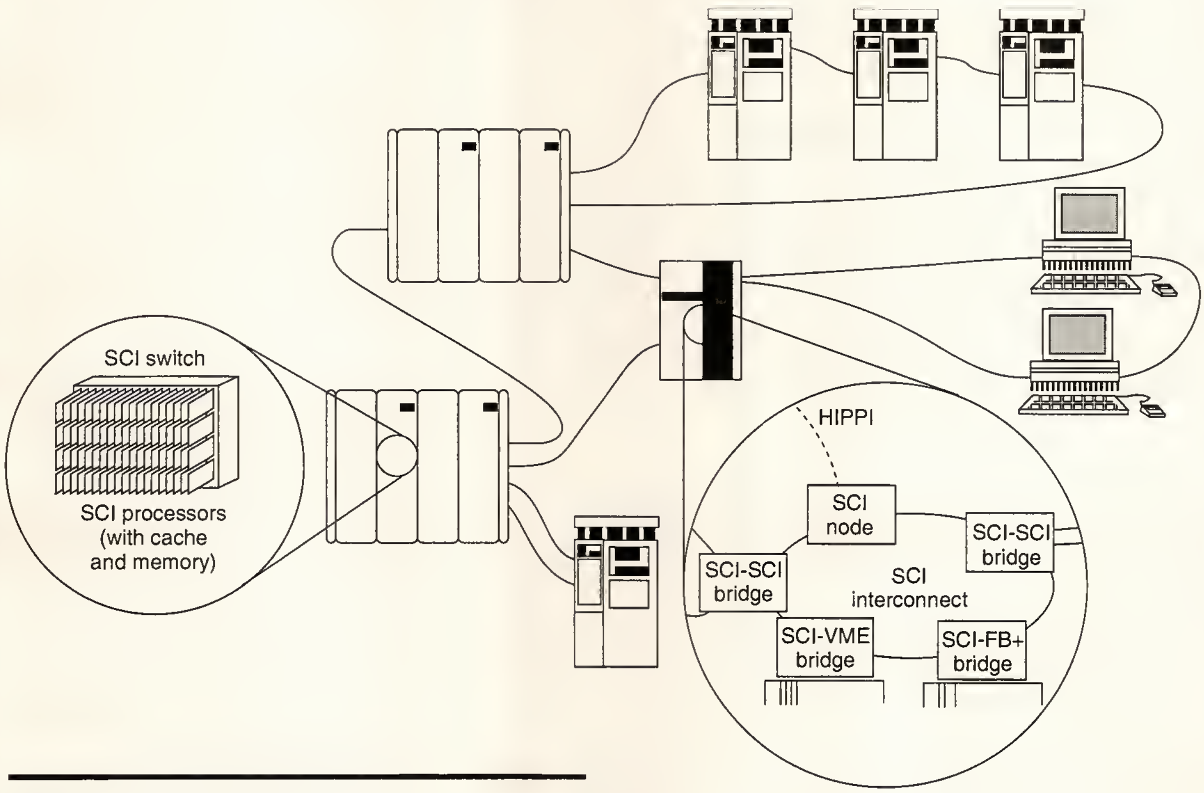


Figure A. A typical SCI application.

A standard SCI module defines signals, connector, and power for operation at a link speed of 1,000 Mbytes/s. A standard cable and connector defines the use of these signals for applications in which the module form factor is not appropriate, over short distances (meters). A standard fiber-optic serial interface solves interface problems over longer distances (kilometers) at a link speed of 1,000 Mbits per second. The same bit stream may be sent over coaxial cable at lower cost for medium distances (tens of meters). We plan to standardize other speeds and signaling in the future as appropriate.

Solving the bus signaling and bottleneck problems. SCI needed two fundamental changes from the way a bus transmits information. First, to make signaling speed independent of the size of the system, interfaces don't wait for each signal to propagate (that is, a bus cycle) before sending the next signal. Each communication takes place by sending packets that include an address, command, and data as needed. While the propagation velocity of a packet is still limited by the speed of light, its rate of information transfer is not. As technology advances, the transfer rate can increase indefinitely. Fortunately, computer scientists know techniques that can compensate for the bad effects of delay (latency), but little can be done to compensate for a transfer rate (bandwidth) that is too small. (See Transaction phases box.)

Secondly, SCI uses multiple signal paths (links) so that multiple independent transfers can take place concurrently. For high performance, designers can use separate links for each processor, memory, or I/O device.

We further refined the SCI link design by applying lessons learned in practical multiprocessor systems. The link signals are differential because differential signals produce the least system ground noise and are least sensitive to noise from other sources. The links are unidirectional because bidirectional links create noise when the drivers are turned off or on to reverse the direction of the link, and because turn-around delays increase with cable length, a scalability problem. The links are fast and narrow because we expect pins will always be relatively expensive.

SCI does not use reverse-direction flow control signals because such mechanisms make the amount of buffer storage needed in the interfaces dependent on cable length. To reach high speeds, we use low-voltage differential signals. SCI initially uses 16-bit-wide, ECL-compatible signals because of the industry experience with and support for that standard. Future links will probably use even lower voltages that are chosen for compatibility with VLSI CMOS or GaAs circuitry.

Thus each SCI interface (node) has (at least) two links, one incoming and one outgoing. The links run continuously, sending idle symbols when no packets are being transmitted, so that the receiver can remain perfectly synchronized at all times, ready for action. SCI packets do not need the prologue that is essential for Ethernet or similar networks.

In a high-performance SCI system, a vendor-dependent switch accepts packets from nodes and routes them to the appropriate other nodes as specified by the address in the packet. SCI does not specify the details of such a switch, because many cost/performance tradeoffs exist.

Lowest cost SCI systems, such as desktop systems, typically will use a ring connection instead of a switch, connecting one node's output link to its neighbor's input. The decision to support rings made the interface circuit more complex because a node may receive a packet intended for some other node and have to pass it along. That process requires some buffer memory and some address recognition logic. However, the result is that one interface definition works over a very wide range of applications, increasing the production volume and lowering the costs for everyone. (See Node interface structure.)

Point-to-point signaling is much easier than bus signaling. This, in combination with SCI's low-current, single-voltage (48V) power dis-

Transaction phases

As seen in Figure B, SCI transactions handshake on a packet basis rather than on a bus cycle basis. A request packet contains all the necessary address, command, and possibly data needed to initiate a transaction. The packet is either accepted and stored in the responder's queues or (if there is insufficient space) is discarded. An echo tells the requester whether it can discard its send packet or must retransmit later because it was not accepted. A similar handshake occurs on the response.

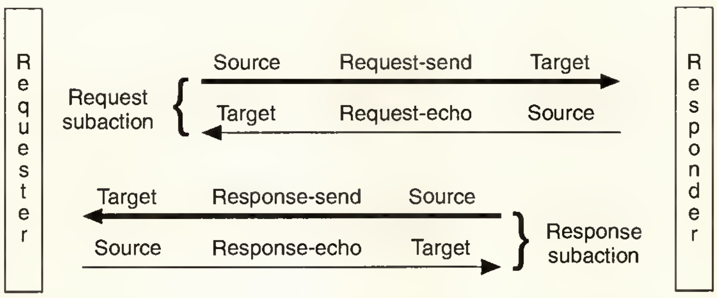


Figure B. Transactions have requests and response subactions.

Node interface structure

Data arriving on the incoming link shown in Figure C have to be resynchronized to the node's own clock. The receiver's elastic buffer circuitry handles this step. The rest of the node circuitry is synchronous, which greatly simplifies the design of the ultrafast FIFOs and logic.

If a node receives a packet intended for some other node while it is transmitting a packet of its own, part or all of the incoming packet moves to the bypass FIFO.

When no packet is being received, idle symbols keep the receiver synchronized and carry information about the priorities of other nodes. They also carry Go bits, which act like tokens and help ensure fair use of the links. Fair use of part of the bandwidth is important in avoiding starvation or deadlock.

An SCI node maintains dual queues to keep responses independent of requests. Without dual queues, excessive requests could prevent the sending of responses, resulting in deadlock.

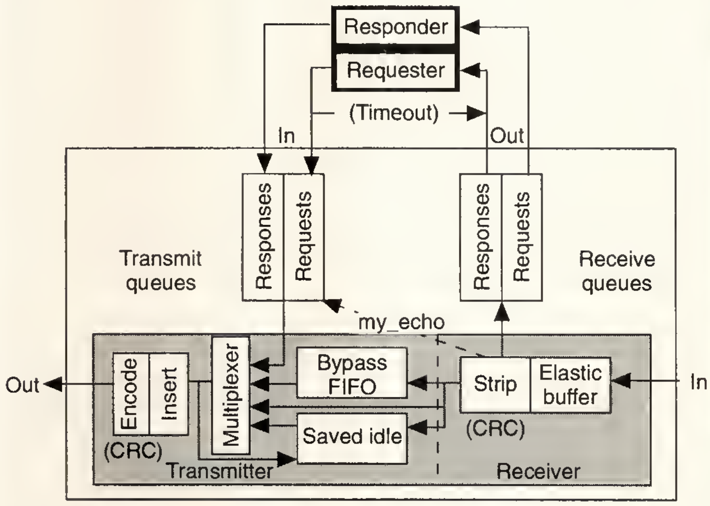


Figure C. Block diagram of a typical SCI node.

tribution system, should make ring-connected backplanes very inexpensive. A few printed-circuit-board layers will generally be enough.

Intermediate-level SCI systems may use a combination of switch and ring, using paired SCI interfaces as a simple bridge that connects two rings. By combining many rings, designers can build a distributed switch fabric. Active research is in progress to discover optimal configurations.[2,3] Scott and Goodman[2] show that when pipelining is permitted, as is the case for SCI, we gain performance rapidly relative to synchronous systems by increasing the dimensionality of the interconnect. The resulting performance is much higher than for packet-synchronous systems.

SCI defines efficient packet-based protocols that provide the kinds of services we expect from a computer bus. The main differences seen by the user relate to the separation of the request for service from the response. A simple bus waits during a memory read access time, until it gets the data. No other parties can use the bus during that wait. This approach makes it conceptually easy to handle error conditions or to perform complex mutual-exclusion operations (like read-modify-write). See the Transaction formats box.

More sophisticated buses split the operation into a request and a response phase, just as SCI does. Then the interface must keep track of pending requests to match the responses to them, and a different style of mutual exclusion becomes necessary. Engineers who have already used split-response

Transaction formats

SCI packets have a 16-byte header that contains address, command, transaction identifier, and (in a response) status information. All packets that may need storage space in queues are multiples of 16 bytes, to simplify storage management at very high speeds. The echo packet is an 8-byte subset of the header (not shown in Figure D); it is never stored in queues. A few transactions need an extended header (not shown either), which adds another 16 bytes.

	Request		Response	
readxx*	Header		Header	0,16,64,256
writexx*	Header	16,64,256	Header	
movexx*	Header	0,16,64,256		
locksb	Header	16	Header	16

* xx represents one of the allowed data block lengths (number of data bytes, on the right after the header).

Figure D. SCI packets.

buses will find SCI to be clean and simple; those who haven't may face a learning curve.

Solving the cache coherence problem. Cache memories are important for keeping processors running at full speed, but they introduce some system management complexity. The various caches often contain duplicate copies of data that must be kept consistent with one another, the cache coherence problem. Say that two processors read the same data (perhaps a synchronization flag) from memory and cache it (perhaps on chip), and then one changes the data (perhaps to free a resource that the other is waiting for). Somehow the other processor must discover that its cached copy is invalid so that it can be updated with current information.

Designers have maintained cache consistency, or coherence, in small bused systems by taking advantage of the bus bottleneck—every cache controller observes every transaction in the system, "snooping" to catch transactions that might invalidate cached data. That approach can be scaled up to a few buses by making the bridges rather sophisticated, but it does not work in highly parallel systems.

A cache coherence scheme that scales to large systems requires a directory that keeps track of which data are being used by which caches, so that the appropriate caches can be updated as necessary. Earlier schemes used a directory but kept it in memory. Instead, SCI maintains it as a distributed, doubly linked list of caches, with the head pointer at a memory controller and the link pointers stored in the cache controllers. With this approach we know that the correct amount of storage is always available for the directory structure, no matter how many caches are sharing copies of a particular line of data. This approach also spreads the maintenance traffic across the system, rather than concentrating it at the memory. Even though these linked-list structures are shared, we designed them to be updated concurrently by multiple processors without any semaphores or lock variables. The protocols use indivisible compare-and-swap transactions instead. (See the Distributed cache tags box.)

We can avoid the cache coherence problem if memory is not shared. Most of the early multiprocessors, which rely on message passing and explicit interprocessor communication, use this approach. However, it is often difficult to move computer programs from single-processor machines to message-passing multiprocessors. Note that shared-memory machines can easily pass messages, so they are more versatile. The

Distributed cache tags

SCI is a distributed system with many links carrying data independently and concurrently. Only directory-based cache coherence is practical in large high-performance systems of this kind. The directory keeps track of which caches are sharing each cache line of data, so that the right caches can be notified when their copy becomes invalid. Rather than centralizing the coherence directory (in RAM), SCI distributes it among those cache controllers that are sharing the data.

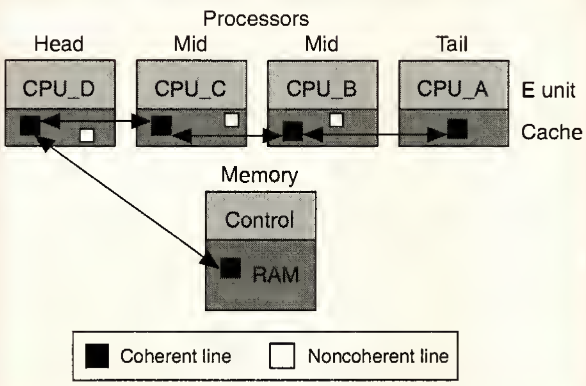


Figure E. A typical linked list for one cache line.

The SCI coherence directory always has storage available in exactly the amount needed, because the cache controllers sharing the given cache line provide it. The arrows in Figure E represent bidirectional pointers forming a doubly linked list. These pointers and a few status bits constitute the directory entry for this cache line.

The doubly linked list structure makes it possible for any cache to roll out a cache line to make room for new data, removing itself from the list by telling its neighbors to point to each other. This list maintenance traffic is distributed throughout the system, greatly reducing the concentration of traffic that is typical of centralized directory schemes.

The protocols support optional optimizations for important cases like pairwise sharing and serial resource allocation. Pairwise sharing lets two sharers pass data back and forth as needed without involving memory. Serial resource allocation (queue on lock bit) uses the list structure to pass the resource along without needless communication in the interconnect.

Note that the directory entry is a shared data structure that may be concurrently accessed by multiple processors. The SCI protocols rely on atomic compare-and-swap operations to ensure correctness without using semaphores or lock variables, which would be less efficient and introduce management complexity. For example, what do you do when a processor sets a lock variable and then dies?

Cache coherence

Until recently, microprocessor designers considered the use of a cache to be their own decision, based on cost/performance objectives, with little concern for the needs of multiprocessors. With just one processor, the existence of a cache usually affects its performance but not the correctness of its execution. A common exception occurs in the case of self-modifying code, where the processor executes instructions from the cache that are writing to memory in a futile effort to modify themselves.

However, in a system with two or more processors each of which has a cache, serious problems can result if the caches are not properly designed to work in harmony. For example, suppose two processors use a variable to keep track of which one is using the printer, and the software waits until no one is using it before starting a new print job. If two primitive processors read the same variable and each keeps it in its own cache, they each read only their cached copy, not any changes that the other processor may make to the variable. Designers must either arrange that such variables are never stored in caches or design the cache control so that each cache discovers when its copy of the variable is no longer correct, discards the old (invalid) value, and obtains a fresh copy.

Such mechanisms add cost and complexity, so it is not surprising that they were omitted in early caching (uni)processors. We call keeping all the cached copies of a data item consistent the cache coherence problem.

The cache controller can determine that its copy is invalid either by tracking every transaction in the system that might be capable of changing the data or by being notified by a reliable source. Bused multiprocessors usually use snooping to maintain coherence. Each cache monitors the bus at all times, which keeps it fairly busy since it also has to be handling the memory requests from its processor. When the cache controller identifies another processor writing to the address of data it has cached, it can either pick up the new value on the fly ("snarfing") or mark the old one invalid and go to memory for a good copy when (and if) the processor needs it later. Say a cache can verify that it has the only good copy—because it changed the data—and it sees another processor trying to read the data from memory. It must then intervene to prevent the other cache from accepting a stale copy. Either the cache can supply the data itself or it can abort the transfer, update memory with the latest copy, and then let the transfer be retried.

The snooping mechanism relies on every controller being able to track every transaction. That's acceptable on a bus, because only one transaction can happen at a time, and each controller tracks it. With effort and care, snooping can even work in a system with several buses. But in a highly parallel system, like SCI, the cost of snooping is intolerable; too many transfers happen at the same time and in different places.

The usual solution to this problem is to maintain a directory that tracks which caches have copies of each cache line. (A cache line is the amount of data tracked as a single entity by the cache controller. SCI tracks 64-byte lines; earlier systems usually used smaller line sizes.)

The next decision is where to keep the directory and

continued on p. 16

goal, then, is to make the coherence protocols efficient and economical.

See the Cache coherence box for further amplification of the problem.

Multiprocessor issues. SCI designers placed a high priority on eliminating several architectural problems, such as deadlocks and livelocks, that have plagued past multiprocessors. Deadlocks can occur when two processors request the same two resources in the opposite order. Each processor may access one resource (blocking the other processor) then become blocked by failure to access the other resource. Livelock, or starvation, occurs when one processor repeatedly accesses a shared resource without letting another have its turn for an arbitrarily long time. Deadlock and livelock are examples of what we generically refer to as "forward progress" issues. To guarantee forward progress, each operation should result in some useful work done, so that the system will not consume all its resources performing useless retries.

Experience with multiprocessor systems shows that they tend to synchronize themselves around problems like these. Even though the designer has estimated the probability of livelock to be extremely small, its effect on the system once it occurs is such that it tends to cause frequent repetitions.

Therefore, we designed the SCI protocol to eliminate dependencies that can cause deadlocks and to include a simple mechanism that assures fair allocation of resources to avoid livelocks. In a few cases, deadlock avoidance seems to make the protocol less efficient (until we consider the indirect consequences!). But in most cases choosing a clean protocol resulted in no penalty.

Of course, software that is outside SCI's scope can still create deadlocks. The SCI designers could only eliminate deadlocks from the underlying protocols.

SCI includes efficient mechanisms for supporting shared

resources, shared data structures, and mutual exclusion. Since the old standby read-modify-write is impractical in a parallel-processing environment, SCI designers took a fresh look at the problem and incorporated several mechanisms that will be helpful.

In a cache-coherent environment, the mechanism that guarantees exclusive use of a cache line by a writer can handle mutual exclusion. The processor can temporarily lock an exclusive cache line while it performs any operations it desires then release it to other readers or writers. However, coherence may not be available in all cases. For example, when accessing (through a bridge) a bus that does not support cache coherence, one may need to tell the bridge to perform a read-modify-write.

Therefore, SCI defines a set of lock primitives that experience shows to be useful for a variety of purposes: masked swap, compare-and-swap, and fetch-and-add. Each of these primitives sends the command and necessary data to a destination device (perhaps a bridge or a memory controller). That device performs the operation indivisibly, returning the result to the requester. This mechanism works well through switches or other interconnects. (It even works well on buses.)

A particularly interesting use of the swap operations is in the maintenance of shared lists that hold information for an interrupt-servicing processor or commands for a DMA controller. If these lists are properly structured, multiple processors can add items to them while one processor takes items off, without any need for lock variables.

A clean way to handle interrupts is to associate a particular shared list with one priority of interrupt service and a specific bit in an interrupt-triggering control register. To request interrupt service, an I/O device or processor adds a work item

Cache coherence *(continued from p. 15)*

how to organize it. One way is to keep somewhere in memory a bit map for each cache line, setting a bit corresponding to a particular cache when that cache comes to the memory for the data. But in a large system, this would require too many bits. Using thousands of bits to account for the location of one cache line isn't acceptable. The next refinement might be to make a list in memory, keeping track of the identifiers of the caches that have taken copies. But while most cache lines are shared by perhaps only one or two caches, some might be shared by every cache in the system. Thus the storage for the worst case list becomes impossibly large. Designers have used a variety of clever compromises, such as allocating a small amount of list storage and then assuming the worst when it runs out. In that case, they assume that every cache in the system has a copy and must be notified when the data changes.

SCI features a very general and scalable directory structure. Memory controllers keep (and store in special memory) one node pointer and a few state bits for each cache line they have. That pointer points to the head of a list. The address of the data in a read transaction routes the read to the memory and tells the memory which cache line is desired. The read transaction includes the node identifier of the requester. The memory controller exchanges that identifier with its pointer for that cache line, returning the old pointer value to the read requester. If the memory has a current copy of the data, it returns the data to the requester too. Otherwise the memory informs the requester that the data is somewhere else, in another cache, and it can use the returned pointer to find it.

SCI cache controllers keep in their tag memory storage, for each line, the memory address of the line (like all cache controllers have to do), a few state bits, and two pointers. The pointers form a doubly linked list of those nodes that have the particular line in their caches.

There are two particularly good properties of this scheme. First, it scales properly. No matter how many caches or how much memory or what the sharing behavior happens to be, exactly the right amount of storage is always available, namely two pointers per cached copy plus one pointer per cache line in memory.

Second, maintaining this list is a distributed process. Only one transaction touches the memory for each request. All the rest (following pointers, adding itself to the list, removing itself when its cache overflows and it needs to roll out a cache line to make more room) involve transactions among the distributed caches, which do not contribute to traffic congestion at the memory.

Though the bus-snooping mechanism may seem conceptually simpler, keep in mind that it bears a high price. It prevents us from having thousands of transactions proceeding at once. Furthermore, every cache in a snooping system must participate in every address cycle. Thus, every cache must be very fast so that participation does not slow the system too much, because the slowest one sets the speed for all.

In the SCI scheme the cache controller acknowledges a packet's arrival and checks it when it is convenient. If the controller is slow, it only slows the transactions in which it participates, not all transactions. As designers add new, higher performance, caches to a system, they get a proportionate performance improvement. This is another desirable kind of scaling behavior; designers don't have to discard old cache controllers to get the benefit of new ones.

Interrupts

Interrupts are simple in a single-processor system, because it is clear which processor should perform the requested service.

Some simple systems just use one signal line to trigger an interrupt. The interrupt causes the processor to save its state on the stack or in a duplicate set of registers, then execute software that asks all possible interrupt sources whether they need service—interrupt-driven polling.

More sophisticated systems allow the interrupting device to identify itself. Some kind of arbitration determines which source has the right to put its identifying vector on the bus during the interrupt-acknowledge cycle. The vector generally changes into an address at which the corresponding service code starts in memory. The arbitration mechanism varies from system to system. It may use a central method, so that individual signal lines connect to each interrupt source, or a distributed method, using a daisy chain or some other arbitration mechanism. Often a combination of polling and vectoring is used. For example, the service routine might poll a list of devices known to be connected to interrupt line 3.

These mechanisms are adequate for finding the source of an interrupt but do not deal at all with the problem of determining which of several processors should service it.

Systems with multiple processors require a more general mechanism. The most common solution provides a special control register associated with the interrupt system of each processor. To generate an interrupt, a device must become the bus master and write to the interrupt register located at the address corresponding to the processor to be interrupted. Becoming a bus master and executing a write may require additional hardware in the interrupting device. So, some multiprocessor systems (Fastbus, IEEE Std 960) also provide a hybrid capability that allows a simple device to assert a signal that causes a shared intermediary device to (possibly poll and then) perform the write for it.

It is tempting to design the interrupt registers to accept vector information directly. A device might write its assigned identifier into the register for use much like the vector used in simple interrupt systems. This temptation should be avoided, because there are hidden perils. For example, what happens if a second interrupt-write arrives soon after the first? Either it must be stored, implying a FIFO, or it has to be aborted and retried later. Any FIFO has a finite capacity, however, so under adverse conditions it might not be capable of holding any more vectors. But aborting the write and retrying also causes trouble, because it can lead to deadlocks. For example, suppose the interrupt service routine of one processor must send an interrupt to another, and the other processor has a service routine that has to interrupt the first. When both processors' FIFOs fill for some reason, deadlock ensues because neither can do anything except continue to retry sending the interrupt.

We can argue that these conditions are anomalous and designing the hardware to have large enough FIFOs will make the problem unlikely to occur. However, experience shows that real multiprocessor systems seem to seek out these trouble spots.

A clean solution puts the burden of allocating resources on the interrupt requester, so that it cannot ask for an interrupt unless it has the resources to ensure the interrupt request can be delivered. Suppose the requester allocates a block of memory to hold its vector and possibly other service information, links the block into a list of service requests, and then writes a pulse to the interrupt service register. Now, no possibility of being blocked exists, and no deadlock can occur.

We can design the service-request list so that any number of interrupt requesters can concurrently add their blocks to the list while one server removes blocks to service them. We will not need semaphores or lock variables, if the system supports indivisible swap and compare-and-swap transactions.[4] These transaction types are extremely valuable in multiprocessor systems and deserve wide support by new processor architectures.

to the list describing what needs to be done, then sets the appropriate bit in the interrupt register. The item in the list is similar to the interrupt vector in single-processor systems. When the processor is ready to service that priority of interrupts, it takes work items off the list and services them.

This mechanism is clean because the service requester allocates all storage, so there is no danger of the server running out of storage when the work piles up (a possible cause of deadlock). The interrupt bit is simple to implement because it is only a latch, with no FIFO storage or critical timing implied. Setting the bit merely alerts the processor that the list should be checked; the processor can clear the bit as soon as it commits to look at the list. (See the Interrupts box for more information.)

A common use of mutual exclusion is to grant sequential use of one resource to a series of requesters. This process can generate a large amount of useless interconnect traffic as processors keep updating their cached copies of the shared ex

clusion variable. SCI defines a queue-on-lock-bit, or QOLB, mechanism that uses the linked lists of the cache coherence system to pass the resource efficiently from one processor to the next.

Statistics on memory sharing are controversial, because few relevant machines exist and because the usage patterns on any real machine will evolve to optimize performance on that architecture. However, most researchers agree that unshared data is the most common case, followed by pairwise shared, followed by multiply shared. Thus the pairwise sharing case may be an important one to optimize. SCI defines optional pairwise sharing optimizations that allow two processors to pass data back and forth without interacting with memory, thus distributing system traffic and reducing activity at the memory controllers.

We made significant changes in the initialization model and in the executable C code that embodies the detailed specifications. In particular, we changed compile-time options to runtime options so that the same code could be used for testing the interaction of nodes implementing different option sets. Also, we significantly improved bit-serial link specifications using material supplied by the Serial HIPPI working group.

The revision process converted all but one of the seven negative votes to affirmative by responding to the concerns expressed. As mentioned earlier, the one remaining negative vote can only be changed by converting the C code to Pascal, which would be unacceptable to the working group.

We redistributed the resulting Draft 2.0 to the balloting body in December 1991 and expect that the standard will receive final IEEE approval early in 1992. Supporting chips should be available within a few months. Dolphin SCI Technology of Oslo, Norway, will offer single-chip SCI interfaces that incorporate transceivers, FIFOs, and cache coherence support. Several versions are planned, for use as a processor, a memory, or an I/O interface. Hewlett Packard is preparing parallel/serial converter chips to interface the Dolphin chips (parallel) to the 1,000-Mbps serial encoding used on optical fiber or coaxial cable. These chips should be available early in 1992.

We expect SCI, Draft 2.0, to receive final IEEE approval early in 1992 and supporting chips to be available within a few months.

Related standards projects

The following projects are either used by SCI or form a part of ongoing work related to future SCI developments.

IEEE Std 1212. The Control and Status Register Architecture standard defines the I/O architecture for SCI, Futurebus+ (IEEE Std 896.1 and 896.2-1991), and Serial Bus (P1394). David V. James, Apple Computer, 20525 Mariani Ave., Cupertino, CA 95014, phone 408-974-1321, fax 408-974-0781, dvj@ apple.com, chaired the group. Voters approved the draft standard, which was then modified in response to ballot comments. The balloting body received a second and final recirculation, and the IEEE Standards Board granted final approval in December 1991.

IEEE Std 1301. The *Metric Equipment Practice for Microcomputers—Coordination Document* is an approved standard (June 1991). This specification defines the generic metric modular packaging family used by the SCI module. Hans Karlsson, Ericsson Telecom AB, TN/ETX/T/F, Stockholm, S-126 25 Sweden, phone +46-8-719-6037, fax +46-8 719 8282, chaired the group.

IEEE Std 1301.1. *The Detailed Standard for a Metric Equipment Practice for Microcomputers Using 2-mm Connectors and Convection Cooling* is also approved (June 1991). This specification details the specific subset of IEEE Std 1301 that is used by the SCI module. Hans Karlsson served as chair. EIA IS-64, February 1991, presently defines the 2-mm connector, but it will become an IEC standard soon. (This connector family is sometimes called Metral, which is DuPont's trademark.)

P1394. The Serial Bus working group is developing a high-speed (10-20 Mbytes/s) serial bus that can be used for low-cost diagnostics (supporting IEEE Std 1149.1 Boundary scan Architecture in large systems) and I/O. The group aims at a very low cost ($15 per connection, including cable, connector, and interface) bus for use in consumer products. SCI includes a Serial Bus connection in the module power connector. Michael Teener, Apple Computer, 3535 Monroe St., Santa Clara, CA 95051; phone 408-974-3521, fax 408-985-9893, teener@apple.com, chairs the group. Work on this standard should complete in 1992.

P1596.1. The SCI/VME Bridge project is defining a bridge architecture for interfacing VMEbuses to an SCI node. This project provides I/O support for early SCI systems via VME. Products are likely to be available in 1992. Bjorn Solberg, CERN, CH-1211 Geneva 23, Switzerland, phone +41-22-767-2677, fax +41-22-782-1820, bsolberg@dsy-srv3.cern.ch, chairs the group.

The project's main decisions involve the mechanism for mapping addresses between VME and SCI, which versions of VME to support, and how to handle interrupts, mutual exclusion, and cache coherence.

P1596.2. The Cache Optimizations for Large Numbers of SCI Processors project is developing request combining, tree-structured coherence directories, and fast data distribution mechanisms needed for systems with thousands of proces-

sors, compatible with the base SCI coherence mechanism. Ross Johnson, Computer Science Dept., 1210 Dayton Street, University of Wisconsin, Madison, WI 53706, phone 608-262-6617, fax 608-262-9777, ross@cs.wisc.edu, chairs this group.

This working group is developing and extending ideas that came up during the development of the base SCI standard, but which it felt could be postponed to avoid delaying SCI's introduction. That is, the protocols defined by P1596 seem adequate for systems with perhaps hundreds of processors (enough for a year or so) and include hooks for adding these optimizations later.

When a large number of requests is addressed to the same node, the interconnect becomes congested and performance suffers. Certain kinds of requests, such as reads and fetch-and-adds, may be combined in the network to reduce this congestion. Request combining allows several requests to be combined into one when they meet (waiting in queues in the interconnect). All but one of these requests generate an immediate response, which tells the requester to get the data from that one's cache instead.

SCI nodes handle this kind of no-data response already, because it is used in the basic coherence protocol. The remaining request goes forward, eventually resulting in a response that provides the data to that cache, where the other nodes read them. This design spreads out the traffic, reducing congestion.

Note that these immediate responses relieve the interconnect from retaining any information about the combining, which enormously simplifies the process compared to previous implementations.

Once the data become available, the time needed to distribute them to all the requesters becomes important. The linear linked lists of the base SCI standard result in times proportional to the number of requesters, which can be a performance problem in large systems.

Instead of linear lists, the group would like to maintain a tree structure, which could distribute the data in time proportional to the logarithm of the number of requesters.

At first, group members thought it would be impractical to maintain binary trees in the distributed coherence directory because the overhead would be too high. The schemes they had seen others use were not acceptable for SCI. These involved setting lock variables to get mutual exclusion while tree maintenance was done, thus scaling poorly and violating an SCI design principle. (Lock variables also introduce a variety of complications, such as what to do when the process that holds the lock fails.)

Thus the group first considered using approximate or temporary pointers, which would form short cuts along the linear directory lists but gradually become inaccurate as processors rolled out cache lines and so on. Whenever a temporary pointer was used, it would be checked for validity, and the algorithm would drop back to following the (always valid) linear list when necessary.

But at the August 1991 meeting, Ross Johnson presented a method for maintaining correct trees at all times, without using lock variables and without adding much overhead. Though details need to be worked out and some corner cases need more study, the group feels the remaining questions can be resolved.

Open issues concern the worst case scenarios (when things happen in the worst possible sequence) and how to reduce the likelihood of having these occur.

P1596.3. The Low-Voltage Differential Signals for SCI project specifies low-voltage differential signals suitable for high-speed communication between CMOS, GaAs, and BiCMOS logic arrays used to implement SCI. The object is to enable low-cost CMOS chips to be used for SCI implementations in workstations and personal computers, at speeds of at least 200 Mbytes/s. Gary Murdock, National Semiconductor, 642 Pineview Drive, San Jose, CA 95117, phone 408-721-7269, fax 408-721-7218, chairs this group.

Five current projects support future SCI applications.

Faster signaling requires smaller signals, if edge rates and currents are to be kept reasonable. Smaller signals require differential signaling (or at least their own reference independent of system ground). At first glance, differential signaling seems to cost a factor of two in signal traces and pins. But the real cost is much smaller because far fewer ground pins are needed, far less system noise is created (or picked up), and the higher signaling speeds reduce the number of parallel signals needed.

SPICE modeling shows that we can signal at SCI speeds (2 ns/bit/signal pair) with contemporary CMOS technology. MOSIS (a fast-turnaround prototype chip fabrication service) test chips have already reached nearly this performance. In fact, the hardest part of the problem is how to provide or accept the data at the signaling rate!

The group bases present modeling on a 250-mV voltage swing, centered on 1V. This approach provides a little headroom for common-mode rejection at the receiver, while allowing use of 5V, 3.3V, and eventually 2V technologies.

The working group is choosing certain signal levels and rates to be supported as signal interchange (link) standards for CMOS implementations of SCI. It will also define an 8-bit and possibly a 4-bit link to complement the Std 1596-defined 16-bit and 1-bit links. This work should complete in 1992.

P1596.4. The High-Bandwidth Memory Chip Interface

As technology advances, SCI will define new physical link standards for higher performance or lower cost.

project defines an interface that will permit access to the large internal bandwidth available inside dynamic memory chips. The goal is to increase the performance and reduce the complexity of memory systems by using SCI signaling technology and a subset of the SCI protocols. This work was started by Hans Wiggers of Hewlett Packard Laboratories, and I now chair it. (My address appears at the end of this article.)

A serious problem with present memory systems is the need to use a large number of memory chips in parallel banks to get the bandwidth needed for today's powerful microprocessors. As the capacity per chip increases, the smallest memory configuration with adequate bandwidth reaches a point where it has an unreasonably large capacity (and needlessly high cost). Furthermore, the increments for expansion are too large. We hope to get much higher bandwidth from far fewer chips by using SCI signaling technology. This approach will lower the entry cost for low-end systems and raise the performance of high-end systems.

Designers are considering several models.[5] One of the most promising uses several RAM chip ringlets attached to a single controller by 8-bit-wide point-to-point links. The name RAM Link is becoming popular for this approach. Details of the signaling are still being worked out, but there seems to be general agreement to use small signal voltages.

Other issues include whether to perform ECC (error checking and correction) in each RAM chip or in the controller. Including it in the RAM chip leaves the details up to the vendor, with possible future technology improvements being incorporated transparently. Including it in the controller is probably the lowest initial cost and gives the system designer the most control. But ECC involves a performance penalty for transmitting the extra information on the links. We could increase link performance by using a 9-bit-wide link, but that would cost pins and power.

Another open question concerns the link protocol to be used. If an SCI bypass FIFO can be incorporated on the RAM chips, we can use a simplified version of the present SCI protocols. If not, we must define a scheduling mechanism that prevents two packets from being transmitted at once. Predictability issues, such as the effects of refresh or ECC soft-error correction on the RAM chip, complicate scheduling. However, in the November 1991 meeting the group proposed a "designated token" scheduling mechanism that looked very promising. Draft 0.11 was presented at the January meeting.

P1596.5. The Shared-Data Formats Optimized for SCI project specifies data formats for efficiently exchanging data between byte-addressable processors on SCI. SCI supports efficient data transfers between heterogeneous workstations within a distributed computing environment. Current systems require conversions among large numbers of vendor- or language-dependent data formats; specifying a single transfer format greatly reduces the complexity of this conversion problem. In addition to simplifying the data-interchange problem, standard data formats provide a framework for the design of future processor instruction sets and language data types. David V. James, Apple Computer, chairs this group.

The specification defines integer and floating-point sizes, formats, and address-alignment constraints. It supports bit fields as subcomponents of a larger byte-addressable integer datum. Work on this project has just begun, but it should not take long to complete, since much of the groundwork has been done earlier in conjunction with development of the CSR Architecture.[6] Draft 0.50 was presented at the January meeting.

Future plans

As technology advances, SCI will need to define new link standards. For example, present fiber optic technology is currently expensive at 1,000 Mbps and prohibitive at higher rates. Yet this situation changes rapidly, and SCI applications in high-definition television would be greatly helped by a bit rate at least double this. We will monitor progress in this area.

Similarly, SCI offers enormous application possibilities for slower links. An 8-bit-wide link operating at 250 Mbytes/s or 500 Mbytes/s might be about right for the next-generation personal computers. Perhaps it will be appropriate soon to define a personal computer form factor and signal standard for SCI, based on new CMOS chips or processors with integrated SCI interfaces.

While designing SCI, we learned a lot about how the processor should interact with the interconnect. We are considering how best to spread this information to processor designers, to make multiprocessor systems more efficient. Possibly, this could be a recommended practice, or even a standard, clean, 64-bit RISC architecture optimized for use with SCI.

We have in mind two projects for bridges to Futurebus+ that are currently waiting for the right time to start. One is a very simple bridge to Profile B, an I/O bus with no cache coherence. The other is a general symmetric bridge that includes cache coherence.

THE BASE SCI STANDARD COVERING THE physical signaling, logical protocols, and cache coherence mechanism should be approved by the IEEE Standards Board in March 1992. The first commercial implementer (Dolphin SCI Technology, Oslo, Norway) expects to have working prototypes within a few months of the standard's approval and has promised to make the interface chips available to others.

SCI's performance seems such a large step ahead of the current state of the art in computer buses that it has some difficulty in appearing credible. The best answer to these doubts will be the existence of working silicon, available at reasonable prices, being used in working systems.

The complexity of SCI is approximately the same as that of a split-cycle bus system (like the VAX BI or Futurebus+ or Fastbus with Buffered Interconnects) in small applications. It is much less than that of a bus system for large applications. Nevertheless, relatively few designers have experience with split-cycle bus design issues, and therefore we foresee some need for training as they move to SCI.

SCI has no evident competition in terms of open systems that could hope to deal with the massive computation needs for the next generation of data acquisition, analysis, and general computation. Nor is there any competitor that spans SCI's whole application range: campuswide optical LAN, desktop workstation bus, network shared server, I/O interface, data acquisition system, highly parallel multiprocessor, supercomputer.

For details, or to participate in this work, please contact me.

Acknowledgments

The US Department of Energy contract DE-AC03-76SF00515 supported this work.

References

1. "SCI-Scalable Coherent Interface, P1596/D2.00 18Nov91," Draft for Recirculation to the Balloting Body, prepared by the P1596 Ballot Review Committee of the IEEE Microprocessor Standards Committee, 1991; copies available from D.B. Gustavson, dbg@slacvm.slac.stanford.edu.
2. S.L. Scott and J.R. Goodman, "Performance of Pipelined K-ary N-cube Networks," Computer Sciences Tech. Report 1010, Univ. of Wisconsin, Madison, Wisc., Feb. 1991.
3. J.W. Bothner and T.I. Hulaas, "Various Interconnects for SCI-Based Systems," *Proc. Open Bus Systems*, VFEA Int'l Trade Association, 10229 No. Scottsdale Road, Suite B, Scottsdale, AZ 85253, 1991, pp.197–202.
4. J.M. Mellor-Crummey, "Concurrent Queues: Practical Fetch-and-Phi Algorithms," Tech. Report 229, Nov. 1987, Univ. of Rochester, Computer Science Dept., Rochester, New York.
5. S. Gjessing, G. Stone, and H. Wiggers, "RamLink: A High Bandwidth Point-to-Point Memory Architecture," *Proc. Compcon Spring 92,* to be published by IEEE Computer Society Press, Los Alamitos, Calif., in 1992.
6. *IEEE Std 1212-1991, Standard for Control and Status Register (CSR) Architecture for Microcomputer Buses;* prepublication copies available from IEEE Service Center, Piscataway, N.J.

Bibliography

K. Alnes et al., "Scalable Coherent Interface," *Proc. Comp Euro 90,* pp. 446–453.

K. Alnes et al., "Chip Sets for Scalable Coherent Interface," *Proc. Open Bus Systems*, VFEA Int'l Trade Association, pp. 209–213.

G. Delp et al., "Memory as a Network Abstraction," *IEEE Network*, July 1991, pp. 34-41.

S. Gjessing, "SCI Cache Coherence," *Proc. Open Bus Systems*, VFEA Int'l Trade Association, pp. 189–196.

S. Gjessing and E. Munthe-Kaas, "Formal Specification of Cache Coherence in a Shared Memory Multiprocessor," Informatics Research Report, Univ. of Oslo, Nov. 1991.

S. Gjessing, S. Krogdahl, and E. Munthe-Kaas, "Formal Specification and Verification of SCI Cache Coherence," *Proc. NIK*, 1989; also available as Informatics Research Report 142, Univ. of Oslo, Norway, 1990.

S. Gjessing, S. Krogdahl, and E. Munthe-Kaas, "Approaching Verification of the SCI Cache Coherence Protocol," Informatics Research Report No. 145, Univ. of Oslo, Aug. 1990.

D.B. Gustavson, "IEEE P1596, A Scalable Coherent Interface for Gigabyte/s Multiprocessor Applications," Special Issue, *Trans. on Nuclear Science,* Vol. 36, No. 1, 1989, pp. 811–812.

D.B. Gustavson, "Scalable Coherent Interface," *Proc. Compcon Spring 89,* IEEE CS Press, pp. 536–538.

D.V. James, "Scalable I/O Architecture for Buses," *Proc. Compcon Spring 89,* IEEE CS Press, pp. 539–544.

D. V. James, "SCI Cache Coherence," *Cache and Interconnect Architectures in Multiprocessors,* M. Dubois and S.S. Thakkar eds., Kluwer Academic Publishers, Boston, pp. 189–208.

D.V. James et al., "New Directions in Scalable Shared-Memory Multiprocessor Architectures: Scalable Coherent Interface," *Computer,* Vol. 23, No. 6, June 1990, pp. 74-77.

E.H. Kristiansen, "Scalable Coherent Interface," *New Backplane Bus Architectures,* CERN report CN/90/4, Mar. 22-23, 1990, pp. 67-75.

E.H. Kristiansen, "SCI-to-VMEbus Bridge, IEEE Std. Project," *Proc. Open Bus Systems*, VFEA Int'l Trade Association, pp. 203–208.

J.M. Mellor-Crummey and M.L. Scott, "Synchronization Without Contention," *Proc. Fourth Int'l Conf. Architectural support for Programming Languages and Operation Systems,* Assoc. of

Computing Machinery, N.Y., pp. 269-278.

J.M Mellor-Crummey and M.L. Scott, "Algorithms for Scalable Synchronization on Shared Memory Microprocessors," *ACM Trans. on Computing Systems*, Vol. 9, No. 1, Feb. 1991, pp. 21-65.

S.L. Scott, "A Cache Coherence Mechanism for Scalable, Shared-Memory Multiprocessors," Computer Sciences Tech. Report 1002, Univ. of Wisconsin, Madison, Feb. 1991.

J.E. Smith and J.R. Goodman, "Restricted Fetch Operations for Parallel Processing by Gurindar S. Sohi," Computer Sciences Tech. Report 922, Univ. of Wisconsin, Madison, Mar. 1990.

P.J. Woest and J.R. Goodman, "An Analysis of Synchronization Mechanisms in Shared-Memory Multiprocessors," Computer Sciences Tech. Report 1005, Univ. of Wisconsin, Madison, Feb. 1991.

David B. Gustavson is a member of the Computation Research Group at the Stanford Linear Accelerator Center in Palo Alto, California. Previously, he was a member of an elementary particle physics research group, also at SLAC, with primary responsibility for real-time data acquisition systems, data processing, and computer interfacing. He chairs the P1596 Scalable Coherent Interface and P1596.4 RAM Link projects. He also chairs the Fastbus Software working group (IEEE Std 1177) and serves as a member of the Fastbus (IEEE Std 960) hardware design team and its executive committee. He has participated in the development of a variety of other standards, including Control and Status Register Architecture (IEEE Std 1212), S-100 bus (IEEE Std 696), Futurebus (IEEE Std 896), and Floating-Point Arithmetic (IEEE Std 754).

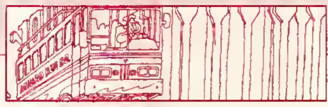
Gustavson received a BS degree in physics and mathematics from the University of Nebraska, studied at the Georg August Universitat in Gottingen, Germany, as a Fulbright Fellow, and received a PhD in high-energy elementary particle physics from Stanford University. He is a member of the IEEE, the IEEE Computer Society, and the Association of Computing Machinery.

Direct questions concerning this article to the author at Stanford Linear Accelerator Center, MS 88, PO Box 4349, Stanford, CA 94309; dbg@slacvm.slac.stanford.edu.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

High 150 Medium 151 Low 152



Unix and the Am29000 Microprocessor

Though targeted for use in medium- to high-performance embedded applications, the Am29000 includes several design provisions allowing its use in Unix workstation applications. For example, a scalable interrupt-handling mechanism makes the processor particularly suited to real-time Unix operations. The relevant features discussed here will help users interested in building a Unix system.

Daniel Mann

Advanced Micro Devices

Use of RISC processors in medium- and high-performance embedded applications is growing rapidly. As prices fall, it seems likely that RISC machines will dominate CISC processors as the system of choice for an even wider range of new embedded system designs. A number of companies are already manufacturing processors using RISC principles that offer a better price-performance ratio than the top-end CISC devices.

Many designers have chosen to use Advanced Micro Device's Am29000 processor in complicated embedded applications. The increased use of higher level languages and real-time operating system support services with embedded RISC applications requires engineers to know more about processor features previously more widely used in workstation applications, such as Unix.

Unix was developed on small computers and it makes modest demands on a host processor. However, for good performance Unix requires the processor to provide certain basic facilities. The Am29000 has several features that make it a particularly suitable Unix host. For example, the architecture is scalable yet has features for maintaining code compatibility.

Our processor's interrupt-handling mechanism services devices without incurring a built-in exception-processing sequence. This is of particular interest to implementers of Unix systems that are also constrained with real-time events. Users are free to design the necessary interrupt-handling processor environment for maximum efficiency.

Among the Am29000's features of particular interest to designers of high-performance Unix systems are

- data and instruction caching,
- memory management,
- multiple data transfer instructions,
- freeze-mode operation,
- multiprocessor support,
- floating-point support, and
- flexible memory systems.

A detailed discussion of the Am29000's architecture[1,2] is beyond the scope of this article, as is a discussion of Unix internals.[3,4] Rather, I point out features that make our processor a good Unix host.

C calling sequence

Making a subroutine call on a processor with general-purpose registers is expensive in terms of time and resources. Because functions must compete for register use, registers must be saved and restored through register-to-memory and memory-to-register operations. For example, a C function call on Motorola's MC68000 processor (see Table 1 on page 24) might use the statements

```
char bits8;
short bits16;

printf("char= %c short=%d", bits8, bits16);
```

After they are compiled, they generate the following assembly-level code:

0272-1732/92/0200-0023$03.00 © 1992 IEEE

Table 1. MC68000 instructions.

Instruction	Comment
MOVE.W saddr,daddr	Move 16 bits of data from saddr to daddr.
MOVE.W -4 [A6], D0	Source address is register indirect with displacement. Destination address is data register direct. The word at memory location –4 relative to the current frame pointer (A6) is copied into data register D0.
MOVE.B saddr, daddr	Move 8 bits of data from saddr to daddr.
MOVE.L saddr, daddr	Move 32 bits of data from saddr to daddr.
EXT.L data_register	Extend the sign of 16-bit data to 32-bit register size.
PEA L15	Push address L15 onto the stack (A7).
JSR _printf	A jump to subroutine _printf is taken. The current PC value is first pushed onto the stack (A7).
LEA 8 [A6], A0	The address of the data object located 8 bytes above the current frame pointer (A6) is loaded into address register A0.
LINK A6,#–32	Push the frame pointer (A6) onto the stack. Then copy the stack pointer (A7) to the frame pointer (A6). The stack pointer (A7) is then lowered by 32 bytes. This instruction takes several cycles and is used in a procedure prologue.
UNLK A6	The frame pointer (A6) is copied to the stack pointer. A new frame pointer is then popped out of the stack. This instruction takes several cycles and is used in a procedure epilogue.
RTS	The address value for a subroutine return is popped out of the stack (A7) and loaded into the PC.

```
L15:  .ascii "char= %c short=%d"

      MOVE.W  –4 [A6], D0   ;stack bits16 variable
      EXT.L   D0
      MOVE.L  D0, –[A7]
      MOVE.B  –1 [A6], D0   ;stack bits8 variable
      EXTB.L  D0
      MOVE.L  D0, –[A7]
      PEA     L15           ;stack text string ptr
      JSR     _printf
      LEA     12 [A7], A7   ;repair stack pointer
```

This assembly listing shows how parameters pass via the stack to the function being called.

The LINK instruction copies the stack pointer A7 to the local frame pointer A6 upon entry to a routine. The parameters passed and local variables in memory are referenced relative to register A6.

To reduce future access delays, the system normally copies data to general-purpose registers before using it. For instance, using a memory-to-memory operation when moving data from the local frame of the function call stack would reduce the number of instructions executed. However, these are CISC instructions that require several machine cycles before completion.

In the example, the C function call passes two variables, bits8 and bits16, to the library function printf( ). The following assembly code shows part of the printf( ) function for the MC68000:

```
_printf:
   LINK  A6, #–32     ;local variable
                        space
   LEA   8 [A6], A0   ;unstack string
                        pointer
   . . .
   UNLK A6
   RTS
```

Several multicycle instructions are required to pass the parameters and establish the function context. Unlike the variable instruction format in the MC68000, the Am29000 has a fixed 32-bit instruction format (see Figure 1). The same C statements compiled for the Am29000 (see Table 2) generate the following assembly code for passing the parameters and establishing the function context:

```
L1:    .ascii   "bits8=%c bits16=%d"

       const    lr2,L1
       consth   lr2,L1
       add      lr3,lr6,0     ;move bits8 and bits16
       add      lr4,lr8,0     ;to bottom of the
                              ;activation record
       call     lr0,_printf   ;return addr in lr0
```

Op code	Operand C	Operand A	Operand B
8 bits	8 bits	8 bits	8 bits

Figure 1. Am29000 fixed 32-bit instruction format.

Register stack. We define a register stack in an assigned area of memory to pass the parameters and allocate working registers to each procedure. The register cache replaces the top part of the register stack, as shown in Figure 2 on page 26.

The global registers rab and rfb point to the top and bottom of the register cache. Global register rsp (also known as gr1) points to the top of the register stack. The register cache, or stack window, moves up and down the register stack as the stack grows and shrinks. Use of the register cache allows data to be accessed through local registers at high speed. On-chip triple-porting (two read ports and one write port) enables the register stack to perform better than a data memory cache, which cannot read and write in the same cycle.

Activation record. Our processor does not apply push or pop instructions to external memory. Instead, each function is allocated an activation record in the register cache at compile time. Activation records hold local variables and parameters passed to the function.

The caller stores its outgoing arguments at the bottom of the activation record. The called function establishes a new activation record below the caller's record. The top of the new record overlaps the bottom of the old record, so the outgoing parameters of the calling function are visible within the called function's activation record.

Although the activation record can be any size within the limits of the physical cache, the compiler will not allocate more than 16 registers to the parameter-passing part of the activation record. Functions that cannot pass all their outgoing parameters in registers must use a memory stack for additional parameters (global register msp points to the top of the memory stack). This happens infrequently, but it is required for the parameters that have their address taken. Data parameters at known addresses cannot be supported in register address space because data addresses always refer to memory and not to registers.

The following code shows part of the printf( ) function for the Am29000 processor:

```
_printf:
        sub     gr1, gr1, 16            ;function prologue
        asgeu   V_SPILL, gr1, rab       ;compare with
                                        ;top of window
        add     lr1, gr1, 36            ;rab is gr126
        . . .
        jmpi    lr0                     ;return
        asleu   V_FILL, lr1, rfb        ;compare with
                                        ;bottom of
                                        ;window gr127
```

The register stack pointer rsp points to the bottom of the current functions activation record. All local registers are referenced relative to rsp. Four new registers are required to support the function call shown, so rsp is decremented 16 bytes.

Table 2. Am29000 instructions.

Instruction	Comment
const reg, value	The value is placed in the selected register. This is a data-direct instruction. Operand fields A and B hold the 16-bit constant.
const lr2, L1	The lower 16 bits of address L1are placed in local register lr2. The high 16 bits of lr2 are cleared.
consth lr2, L1	The high 16 bits of address L1 are placed in the high 16 bits of local register lr2.
add des, srcA, srcB	The two source operands A and B are added, and the result placed in the destination register.
add lr3, lr6, 0	Zero is added to the lr6 value, and the result is placed in local register lr3. This is effectively a register-to-register move operation.
call lr0, _printf	A jump to subroutine _printf occurs in the cycle following the current cycle. The current PC address plus 8 is placed in local register lr0. This is effectively a subroutine call with the destination address obtained by adding the current PC value and the 16-bit value formed by operand fields A and B. The Am29000 implements delay-slot branching, thus it always executes the instruction following a branch instruction. Direct 32-bit address procedure calls are supported with the CALLI instruction.
jmpi lr0	The value in local register lr0 is loaded in the PC. Execution at the destination of a jump commences after the instruction following the JMPI has executed. This instruction implements a subroutine return.
asgeu V_SPILL, gr1,rab	This is a conditional trap instruction. Such instructions are used in procedure prologue and epilogues to check whether cache spilling or filling is required. Trap number V_SPILL is taken if the assertion that the global register gr1 is greater than or equal to global register rab is false.

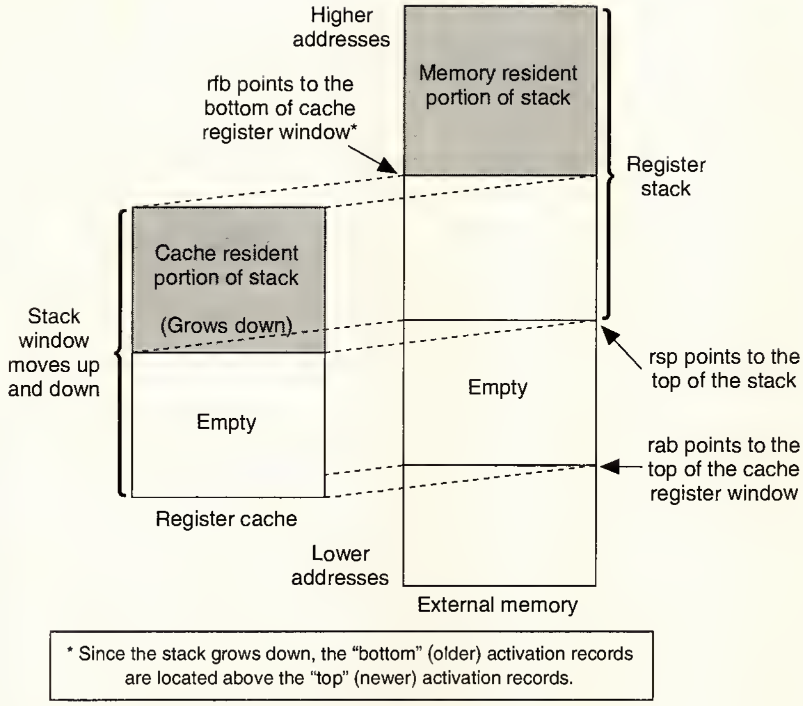


Figure 2. Register stack window.

Rsp performs a role similar to the MC68000's A7 and A6 registers except that it points to data in high-speed registers, not data in external memory.

The compiler reserves local registers lr1 and lr0 for special duties within each activation record. Lr0 contains the execution starting address when it returns to the caller's activation record. Lr1 points to the top of the caller's activation record. The new frame allocates local registers lr2 and lr3 to hold printf function local variables.

As Figure 3 shows, the positions of five registers overlap. The three printf( ) parameters enter from lr2, lr3, and lr4 of the caller's activation record and appear as lr6, lr7, and lr8 of the printf( ) function activation record.

Spill and fill. If not enough registers are available in the cache when it moves down the register stack, then a V_SPILL trap is taken, and the registers spill out of cache into memory. Only procedure calls that require more registers than currently are available in the cache suffer this overhead.

Once a spill occurs, a fill (V_FILL trap) can be expected at a later time. The fill doesn't happen when the function call causing the spill returns, but rather when some earlier function that requires data held in a previous activation record (just below the cache window) returns. Just before a function returns, the lr1 register, which points to the top of the caller's activation record, is compared with the pointer to the bottom of the cache window (rfb). If the activation record is not stored completely in the cache, then the fill overhead occurs.

Performance. The register stack improves the performance of call operations because most calls proceed without memory access. The register cache contains 128 registers, so very few function calls or returns require register spilling or filling.

Because most of the data required by a function reside in local registers, there is no need for elaborate memory-addressing modes, which increase access latency. The function call overhead in the Am29000 consists of a small number of single-cycle instructions; the MC68000 requires a greater number of multicycle instructions.

Context switching

Context switching occurs when one process gives up control of the CPU without terminating, and another process, which had previously given up control, resumes executing. When this happens, the state of the processor being used by the process (the context) must be saved, and the context of the other process must

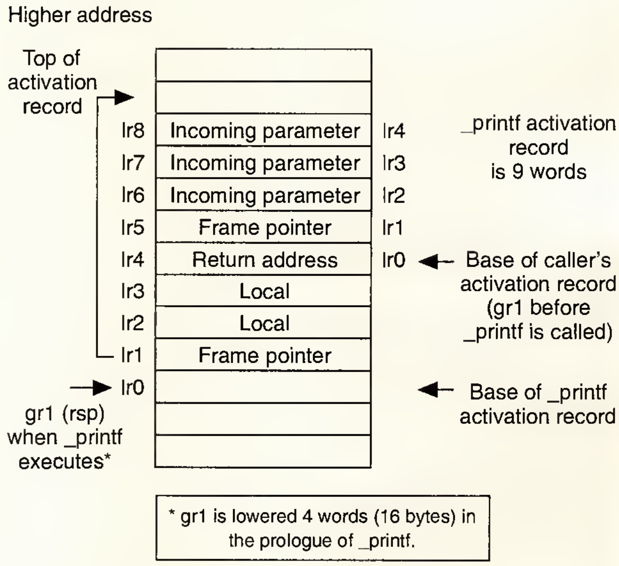


Figure 3. Overlapping registers.

be restored. Other than allowing multiple processes to share the same CPU, this saving and restoring of context perform no useful work. If context switching is performed frequently, it should be a low-overhead operation.

One result of having a large register file is an increased context-switching time. Approximately half of the 128 local registers and about 32 of the global registers contain data for the user's process. About 96 memory writes and 44 memory reads are required to store and reload these registers with a new context. However, with single-cycle, burst-mode access, the register save and restore time can be as short as 5.6 μs. Note that the context reloading for synchronously saved process states is faster than context saving because only the activation record of the currently active procedure need be restored. The number of local registers requiring restoring decreases from approximately 64 to, typically, 12.

Memory access time dominates the total context switch time. In a low-cost system with no external data cache, this is most apparent. When using five-cycle read/write memory, the memory access time is 28 μs. In practice, any increase in context-switching time due to the large register set is more than offset by the savings in normal execution. This is because context switches (30 to 60/s) are less frequent than system calls (150 to 250/s) or subroutine calls (about 350,000/s). The Am29000 can switch context in as little as 10 μs, about 45 times faster than the VAX 11/780.

System calls

System calls are trusted library functions that execute with kernel mode permissions to access resources otherwise unavailable to the user. To users, a system call looks like a normal C function call. But when a user program executes a system call, it leaves user mode and enters kernel mode.

Permission levels. When a system call is made, the selected function executes a trap instruction to change the processor status. After the trap is taken, the process operates with kernel mode permissions. It is usually necessary for the operating system to copy the calling parameters from the user mode register stack to the kernel data space, because system call functions receive their parameters from kernel data memory space. To enable system functions to appear as ordinary C functions, the register stack parameters are transferred into a kernel data structure known as the *upage* by a system call dispatcher routine.

The kernel mode permissions of system calls may not access data that the user would not normally be able to access. When moving data between user space and kernel space, many processors use assembly level instructions to manipulate the memory management unit (MMU) or other protection hardware. This allows the kernel to access the user data space with the user's permissions rather than kernel's permissions.

The usual set of Unix functions available includes a fetch user byte, fubyte(a), and a store user byte, subyte(a,v). As an example, the statement

```
result = fubyte(addr);
```

fetches a byte of data from user address addr and stores it in the variable result. This variable is in kernel data space, and the access of the user data space occurs with the user's permissions. The subyte(a,v) function moves data in the opposite direction, from kernel space to user space. This routine is required to modify data structures in the user's address space.

Protection boundaries. Our processor efficiently copies data across protection boundaries in a Unix implementation. System call parameters are not located in the user's data memory space but in the register cache, where they don't result in a heavy overhead.

After the system call trap is taken, call parameters can be copied from the registers into the upage without any permission problems because the register cache is wholly owned by the user process. There is no risk of the kernel accessing data belonging to another user. The store multiple instruction moves data quickly from registers to memory.

The processor handles system call return values the same way as any other function call return values. Global registers from gr96 through gr111 hold the first 16 words of a function return value. Any other data in the user's data space that the system call uses or updates can be accessed by setting the user access (UA) bit in load and store (LOAD, STORE) instructions. The UA bit allows programs executing in kernel mode to emulate user mode access. Doing so enables functions such as subyte( ) to be implemented inexpensively, particularly since the UA bit can be used with the load and store multiple (LOADM, STOREM) instructions.

Overhead. Because few accesses to data memory are required, the overhead of switching from user mode to kernel mode is small. In kernel mode, separate memory and register stacks are used, but they are still accessed via global registers rab, rfb, msp, and rsp. Upon entering kernel mode, these registers contain user mode values which are then stored in the upage (part of the context save) and replaced with addresses of kernel-mode memory and register stacks. The register stack is not pushed to memory, but rather the kernel register stack is added to the end of the cache.

If the barriers between user and kernel modes are high and secure, the computation required for a mode change may far exceed the cost of the actual requested operation. In cases of extremely poor design, the simplest system calls may require hundreds or even thousands of overhead instructions.

In keeping with RISC concepts, we structured the system call operation to ensure the minimum overheads at each stage. As an example of the results achieved, getpid( ) executes in about 10 μs, depending on the memory architecture. The same function call has been measured at 400 μs on a VAX 11/780. The system call mechanism results in an overhead that is one

hundredth of that incurred by a VAX 11/780.

Interrupt handling

As a RISC processor, the Am29000 limits the mechanism for handling interrupts and traps so system designers need not be concerned about when, where, or how much state is saved. State saving is not built into the processor interrupt mechanism but is left for the programmer to implement. Interrupts use a vector table to select the required service routine. After nine cycles (0.36 μs) or less, the processor executes the service code. A freeze mode makes this possible.

Freeze mode. When an interrupt or trap occurs, the processor enters freeze mode. The hardware execution context described by critical CPU registers is not saved, but the state of these registers is frozen and cannot be updated until an interrupt return is taken or freeze mode is turned off. Interrupts that do not require the use of processor special registers can be serviced in freeze mode. This allows a fast interrupt service response. In fact, it is practical to implement a software cache-controlled reload in low-cost systems.

Certain critical registers are required for the execution of C code routines. If the interrupt-handling code is just assembly glue used to reach a C code routine, then the critical registers must be saved explicitly before freeze mode is turned off. Users can tailor the form of the register context stacking procedure to meet special needs. This facility can prove useful in optimizing system operation.

Interrupt servicing. Our processor's high-speed operation enables interrupts to be handled by a Unix implementation without assigning a priority order. When an interrupt is detected, it will be serviced immediately if no other interrupts are currently being serviced. Otherwise, the interrupt is added to the end of a queue. The processor continually tries to empty this queue and return to executing user processes.

When the first interrupt occurs, possibly initiating a queue, the processor may be in user mode or kernel mode. Although each process executes the same kernel code, each process has its own kernel stack for kernel mode function calls and data. In user mode, the kernel memory stack stores data during the interrupt service routines. If the interrupt occurs in kernel mode, then a separate shared interrupt stack is used.

Although I have described only one stack, the processor contains two: one for conventional memory data and one for register data that has spilled out or swapped out of the register cache.

Virtual address space

Unix is a multiprocess system, with several processes existing in different areas of system memory at the same time. Programs may be loaded at some memory location and then change location during execution due to being swapped out and back in again. Programs are usually intended to execute from virtual address zero, but the actual physical memory used by the program is dependent on the relocation function of the MMU. The program, or parts of the program, may get swapped to new physical locations. But because of the address translation of the MMU, they always appear to be at the same virtual address.

On-chip MMU. We built a standard MMU into the Am29000 chip to lower system cost and maintain compatibility among 29000 users. Most other processors require expensive external hardware to translate addresses. In such cases, designers sometimes prefer to develop their own MMUs. This can lead to incompatibilities with code developed for other designs.

Translations. The Am29000 uses a 64-entry translation look-aside buffer (TLB) to perform address translations. The TLB translates the most frequently used instruction and data memory pages and reflects the information contained in more extensive tables in memory. Data addresses are translated during each execute processor stage. Instruction addresses are translated whenever a jump is taken, achieving virtual memory support without any access delays.

Because of the TLB's limited size, there is a chance that an access will be requested for a memory page not currently translated—a TLB miss. When this occurs, the special register lru (least-recently used TLB entry) selects a TLB register for replacement with new translation data.

The TLB registers are not updated automatically by hardware. When a TLB miss occurs, a trap to a software routine that maintains TLB entries is performed. Using this method to update the TLB allows a variety of memory management techniques to be implemented. Systems that rely on hardware to update the TLB registers do not achieve this level of flexibility. The technique is particularly well suited to the emulation of missing hardware accessed in virtual address space. A TLB register can be updated in two Am29000 cycles (one for each word of a TLB entry). This keeps down the software overhead of reloading TLB. Depending on page-table layout, a TLB reload needs 25 to 40 cycles after a page miss trap occurs.

Memory access protection

In any multiprogramming system, especially one in which users are developing, debugging, and testing software, process *a* should be forbidden access to the data of process *b* if accessing that data would impair process *b*. In addition, some operating system data should be protected even from read-only access.

Reducing memory usage and permitting efficient multiprocess cooperation requires controlled sharing of regions of memory and other resources. For many programs, a large portion of executable code consists of the system-provided library subroutines. For small programs, the library code used by the program can be larger than the program itself, particularly if the program makes heavy use of standard I/O routines. Enhanced Unix systems provide shared libraries so

only one copy of the library code is required for all the processes in the system.

Protection violation checking. The TLB hardware checks for protection violation. Each TLB entry contains a 6-bit field that controls access to the associated page, which is assigned to a particular user through a task identifier. Permissions can be set to separately enable reading, writing, and execution of page data. The processor mode and user ID are checked with the access permissions for the virtual memory page accessed. A software trap occurs if access is not permitted. Including user identifiers in each TLB entry enables Unix to switch processes without clearing all TLB entries. Performance is improved because valid TLB entries are likely to remain and be reused when their associated process is swapped back in.

Protection control. The Am29000 offers considerable access control within the chip. Other systems that use external hardware to achieve a similar function incur a greater system cost. Also, by including the necessary hardware in our design, we achieve a standard that cannot be ensured when designers build their own protection systems.

Cache support

The Am29000 incorporates an on-chip branch target cache (BTC) memory. If the target instructions of a branch are found in the cache, the branch executes in one cycle, causing no stalling of the processor pipeline. For sequential instruction access, an instruction prefetch buffer ensures that instructions are fetched up to four cycles ahead of their execution, enabling single-cycle instruction execution at high speeds.

The BTC memory contains the target instruction sequence for 32 branches. Simulation results show this is sufficient for 60 percent of the branch instructions. If a BTC memory miss occurs, the hardware automatically updates the cache with the new target instruction sequence. In such cases, the execution pipeline stalls for one cycle plus the number of cycles required to access the instruction memory. Having an on-chip instruction cache enables users to implement low-cost systems that achieve high speeds.

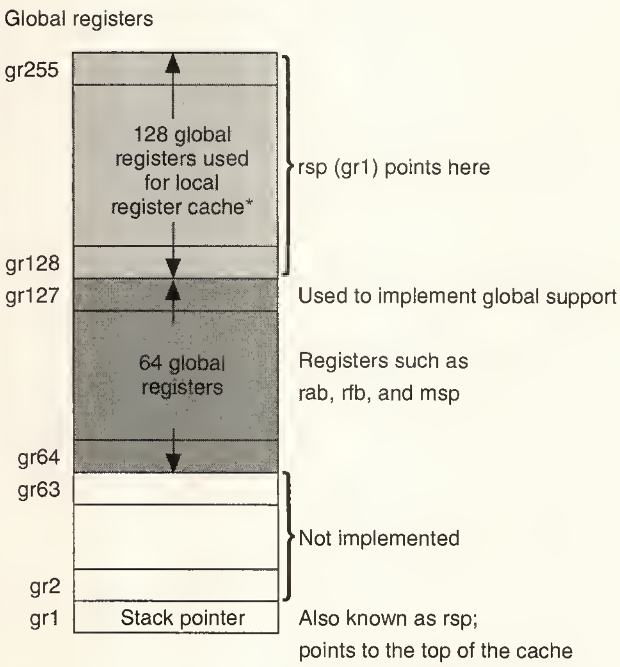
Data caching. The Am29000 processor caches data via a large register file to reduce the number of memory loads and stores. This reduces the requirement for an external data cache. Of the 192 general-purpose registers, 128 serve as a register cache. (See Figure 4.) This cache stores data variables and function call parameters by using an overlapping stack frame technique.

On occasion the cache becomes full and some data spills out to memory. Simulation results of programs such as nroff show this is necessary in only 0.1 percent to 0.5 percent of all calls. Since calls typically constitute 1.2 percent of all instructions used, and spilling out and filling back typically require only 25 to 30 cycles, the overhead is low.

Accessing data in the register file does not suffer the delays encountered in accessing external data, even if a high-speed cache is used. Thus the Am29000 processor is well suited for even higher speed devices in future technologies. The register file has a further advantage in that it is triple-ported. It accesses two sources in one cycle, while a previously computed result is written back. Subsequent accesses take one cycle, using burst mode.

On-chip instruction caching. BTC memory caches user mode and kernel mode instructions. The cache tag logic works with physical addresses, possibly after they have gone through the virtual-to-physical address translation service of the TLB. The BTC memory entries need not be invalidated on a context switch. Keeping entries valid for future reuse improves performance. However, when the kernel decides to swap user process pages in or out of disk memory, instructions at cached physical memory locations may change, invalidating the cache.

The newer Am29030 processor contains an 8-Kbyte instruction cache. The addition of the cache was made possible by the higher levels of integration not possible when the Am29000 was introduced. Research shows that for cache sizes above 4 Kbytes, a conventional cache provides superior performance to the BTC memory method.[5]



* Registers in the cache are normally referred to as local registers and are accessed relative to gr1. Gr1 points to local register lr0. Local register lr1 is located at register address gr1 + 4. All registers are 32 bits wide.

Figure 4. Am29000 register stack.

Because the BTC memory reduces initial instruction access latency rather than improving memory access bandwidth, the Am29000 uses a separate bus to support concurrent instruction fetching. The Am29030 does not require separate buses for instruction and data memory because the on-chip cache offers a statistical improvement to the memory bandwidth. It maintains sufficient instruction memory bandwidth without the support of a dedicated bus.

Multiprocessor Unix

Our processor supports on-chip multiprocessor systems without the need for large amounts of external hardware. It also supports the mechanisms to ensure processor resource sharing and signaling.

An external coprocessor interface extends its execution unit with special-purpose instructions. If the coprocessor hardware is not present, then a coprocessor exception trap is taken. The processor emulates the coprocessor operation for hardware development and code compatibility.

Multiprocessor cache support. Each TLB entry contains 2 bits to control external cache operation. Cached pages can be marked selectively as not-to-be-cached, local, or shared. This enables the operating system to inform the cache of the correct action to take, and it results in reduced data traffic on the shared-memory bus giving access to the cache.

A load and lock (LOADL) instruction for device and memory interlocks activates the LOCK output pin during the address cycle of the access. Multiprocessor hardware uses the lock signal to delay access of other processors requesting the same memory or I/O facility.

The load and set (LOADSET) instruction implements a binary semaphore. The operation of writing a memory semaphore location to True after reading the location into a general-purpose register, cannot be interrupted.

Floating-point support

Unix is a product of the academic and scientific environment, which means that a number of its user application programs require floating-point support. The newer Am29050 processor supports floating-point instructions directly. However, the Am29000 processor instruction set contains a number of floating-point instructions that are not directly supported.

When a floating-point instruction is encountered, it traps to a handler that sends the appropriate instruction and data to an optional coprocessor, the Am29027, and moves the results back to the Am29000. Without the coprocessor, the Am29000 calls software routines to perform the instruction evaluation. This latter method is much slower than using a coprocessor. But by using the trap method, code developed for the Am29000, which may or may not have an Am29027 coprocessor available, will run directly on future Am29000 processors (and at much greater speed).

A library of software routines has been developed that makes extensive use of the floating-point instructions to perform functions. We reserved 15 user-mode-accessible global registers for math library use to achieve good performance for these functions. We can reduce the number of global registers by substituting memory locations, but doing so reduces performance. Also, all library users must agree on register assignment if the library is to be shared.

System costs

The on-chip MMU functions enable the Am29000 to operate without caches and maintain good speed. Table 3 draws on information in the "Am29000 Performance Analysis" document.[6] The table compares AMD's 29000 processors running at 25 MHz against Sun processors and the VAX 11/780.

The benchmarks for the Am29000 in Table 4 are for systems with separate instruction and data memory systems. The Am29000 has separate instruction and data memory buses and a shared address bus. The need to implement a memory system for instructions and a separate system for data can be avoided by connecting the instruction and data buses. Doing so makes the Am29000 processor bus system appear more like a conventional processor bus system. However, sharing a single memory system reduces performance by, typically, 15 percent. The use of video memory (video DRAM) also avoids the need for two memory systems, as the video-out port can be connected to the instruction memory bus, and the random access port to the data bus.

Table 4 gives the approximate costs of a CPU and memory devices. It also shows the cost for a memory system based on 4 Mbytes of DRAM with an additional 0.5-Mbyte SRAM used to implement a cache. Studies show that software-controlled caches can efficiently use high-speed memory with a minimum of hardware complexity and cost.[7] The lightweight interrupt-handling capability achieved by the Am29000 processor operating in freeze mode is well suited to implementing a soft cache control mechanism.

Table 3. CPU speeds.

	Am29000		Am29030	Other processors		
Benchmark	SRAM	DRAM	DRAM	Sun 4	Sun 3	VAX
diff	20.35	12.02	20.86	8.50	3.56	1.0
grep	14.81	9.42	18.54	7.00	3.00	1.0
nroff	14.92	10.78	18.31	7.50	2.30	1.0
Dhrystone 2.0	23.00	15.30	19.69	11.71	2.65	1.0

Table 4. CPU and memory costs.

Feature	Am29000 SRAM	Am29000 DRAM	Am29000 Soft cache	Am29000 Video DRAM	Am29000 DRAM	Am29030 DRAM
Instruction memory	4 Mbytes	4 Mbytes	4-Mbyte DRAM, 0.5-Mbyte SRAM	4 Mbytes	4 Mbytes	4 Mbytes
Data memory	4 Mbytes	4 Mbytes	Shared	Shared	Shared	Shared
Cost	$2,792	$424	$352	$448	$264	$262

The Am29030, unlike the Am29000, shares a bus for instruction and data memory and thus does not require separate memory systems. It achieves higher performance via its on-chip, 8-Kbyte instruction memory cache. The Am29050, which is currently the only 29K family member that directly executes floating-point instructions, is pin compatible with the Am29000 processor and thus supports the three-bus architecture.

Because our processor uses only one address bus, the number of pins and the system costs are reduced. The one-address bus technique is successful because of burst-mode addressing and the large register cache. The register file greatly reduces the need to access data in external memory. Burst-mode addressing enables instructions to be fetched in sequence without sending a new address for each access. The address bus is only required to establish the address of the first instruction in a nonsequential instruction fetch. Video DRAM or conventional RAM interleaved with a small amount of support circuitry, such as an address counter, can be used to implement burst-mode memory addressing.

Contention for the address bus for both instruction and data access is rare. By the very nature of the instruction stream, a jump instruction cannot occur on the same cycle as a load or a store operation, thus the address bus is inherently used for instruction and data addressing on different cycles.

THE AM29000 PROCESSOR'S REDUCED INSTRUCTION set results in concise, efficient executable code. Each processor in this family sustains a performance level of 17 VAX MIPS or higher. In addition, the processors have several design features that make them particularly suitable for efficient Unix implementation, both in terms of operating speed and support peripherals required.

Among these is a scalable methodology that frees the system implementer from a fixed-price, fixed-architecture approach. Because the architectural features required to support Unix are packaged along with the processor, the Am29000 processor is particularly suited to use in low-cost systems.

References

1. *The Am29000 User's Manual*, Advanced Micro Devices, Sunnyvale, Calif., 1989.
2. M. Johnson, "System Considerations in the Design of the Am29000," *IEEE Micro*, Vol. 7, No. 4, Aug. 1987, p. 28-41.
3. M.J. Bach, *The Design of the Unix Operating System*, Prentice-Hall, Englewood Cliffs, N.J., 1986.
4. S.J. Leffler et al., *The Design and Implementation of the 4.3 BSD Unix Operating System*, Addison-Wesley, Reading, Mass., 1989.
5. M.D. Hill, "Aspects of Cache Memory and Instruction Buffer Performance," PhD Report UCB/CSD 87/381, Electrical Engineering and Computer Science Dept., Univ. of Calif., Berkeley, Calif., Nov. 1987.
6. T. Olsen, "Am29000 Performance Analysis," technical report 10621A, AMD, 1988.
7. D.R. Cheriton et al., "Software-Controlled Caches in the VMP Multiprocessor," report STAN-CS-86-1105, Dept. of Computer Science, Stanford Univ., Stanford, Calif., Mar. 1986.

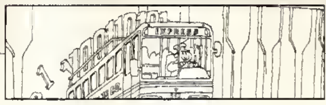
Daniel Mann is a senior member of the technical staff at Advanced Micro Devices' Am29000 Support Products Division. He works on operating systems and debugger issues. Mann earned BS and PhD degrees in electronic engineering from RGIT in Aberdeen, Scotland. He is a member of the IEEE Computer Society.

Address questions concerning this article to the author at Advanced Micro Devices, 5204 E. Ben White Blvd., MS 573, Austin, TX 78741, or via e-mail at daniel.mann@amd.com.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 153   Medium 154   High 155



Hardware Requirements for Neural Network Pattern Classifiers

A Case Study and Implementation

A special-purpose chip, optimized for computational needs of neural networks, performs over 2,000 multiplications and additions simultaneously. Its data path is suitable particularly for the convolutional architectures typical in pattern classification networks but can also be configured for fully connected or feedback topologies. A development system permits rapid prototyping of new applications and analysis of the impact of the specialized hardware on system performance. We demonstrate the power and flexibility of the processor with a neural network for handwritten character recognition containing over 133,000 connections.

Bernhard E. Boser

Eduard Sackinger

Jane Bromley

Yann leCun

Lawrence D. Jackel

AT&T Bell Laboratories

Neural networks are rapidly gaining acceptance as powerful and versatile tools for pattern classification.[1,2] However, the widespread use of neural network classifiers remains contingent on the availability of powerful hardware to provide adequate speed. Such hardware is particularly important as the computational requirements of neural network algorithms are quite different from the high-precision processing for which general-purpose computers are optimized. Typical neural network problems involve huge amounts of low-resolution and possibly redundant data and require a correspondingly high number of low-precision arithmetic operations to be performed.

Special-purpose VLSI (very large-scale integration) processors let us overcome neural network implementation problems. With their regular structure and the small number of well-defined arithmetic operations, these networks are well matched to integrated circuit technology. The high density of modern technologies lets us implement a large number of identical, concurrently operating processors on one chip, thus exploiting the inherent parallelism of neural networks. The regularity of neural networks and the small number of well-defined arithmetic operations used by neural algorithms greatly simplify the design and layout of VLSI circuits.

But processing speed is not the only constraint on neural network hardware design; neural network classifiers benefit from a highly structured topology with local receptive fields. Of particular importance are convolutional architectures in which neurons with identical weights process different parts of the input or internal state. This topology builds into the network knowledge about locality of data, improving recognition performance. At the same time it lets us multiplex neurons with identical sizes and realize the large networks required for difficult classification tasks within the density limitations of current VLSI technology. The high speed of VLSI technology—five orders of magnitude greater than that of

0272-1732/92/0200-0032$03.00

natural neurons—compensates for the loss of computing speed resulting from partial serial processing in such an implementation.

Matching the arithmetic precision of the hardware to the requirements of neural networks is crucial for an efficient hardware implementation. Neural networks are often quoted for their low-resolution requirement, which lends itself well to analog implementation. Experiments, however, show that the precision requirements of neurons within a single network vary. Specifically, research indicates that higher accuracy is often needed in the output layer, for example, for selective rejection of ambiguous or otherwise unclassifiable patterns. This situation can be handled with a hybrid architecture, which evaluates the bulk of the network with low-resolution analog hardware but implements selected connections on a digital processor with higher accuracy.

Based on these considerations, we designed and fabricated ANNA (Artificial Neural Network ALU), a special-purpose chip for neural network pattern classification.[3] The chip features 4,096 individual synapses that can be multiplexed to implement networks with several hundred thousand connections. We can program the number of synapses per neuron to values between 16 and 256; the number of neurons varies accordingly between 256 and 16. Implementable network topologies include fully connected architectures with or without feedback, local receptive fields, and TDNNs (time-delay neural networks) or higher order convolutional connections. Depending on the network topology, the chip can sustain a performance of up to 5×10^9 connections per second (C/s). Arithmetic operations take place with 6-bit resolution on the weights and 3-bit resolution on the states.

Because of the high speed, parallelism, low resolution, and other characteristics of ANNA, which differ considerably from those of general-purpose computers, this hardware must be readily accessible to the network designer, so new applications can be prototyped easily. A development system, consisting of a PC or VME board containing an ANNA chip and a digital signal processor (DSP), addresses these needs. We download the topology and weights of a network into the VME board, and the DSP issues control commands to ANNA. The DSP also preprocesses and trains the networks and processes operations that require higher precision than that of ANNA.

We selected an optical character recognition neural network to test and demonstrate the flexibility and power of the neural network chip.[1] This network identifies handwritten digits from a 20×20-pixel input image and employs neurons with local receptive fields as well as a fully connected layer. The network with over 133,000 connections fits on one ANNA and is evaluated at a rate in excess of 1,000 characters per second, which constitutes a speedup of two orders of magnitude over a DSP-based implementation. Despite the low resolution of the chip, the error rates of the neural network processor and DSP implementation are very similar at 5.3 percent and 4.9 percent. For comparison, the measured human performance on the same database is 2.5 percent errors.

Our OCR neural network with 133,000 connections identifies handwritten digits at 1,000 cps and fits on one ANNA chip.

Neural network hardware

Artificial neural networks that solve difficult problems in areas such as speech recognition and synthesis, or pattern classification, consist of thousands of neurons with tens or hundreds of inputs each. Every neuron computes a weighted sum of its inputs and applies a nonlinear function to its result. Architectural parameters, such as the number of inputs per neuron, and each neuron's connectivity vary considerably within a network, and from application to application. A special-purpose neural network processor must be flexible and powerful enough to accommodate a wide range of applications. At the same time, the requirements must be carefully balanced and the special nature of the task exploited to bring an efficient implementation within reach of today's technology.

We can distinguish two phases of operation in many neural network applications. During the learning phase, the topology and weights of the network are determined from a labeled set of examples using a rule such as backpropagation,[4] or a network-growing algorithm.[5] In the subsequent retrieval or classification phase, the network parameters are fixed.

The network recognizes patterns based on information stored in the architecture and weights during training. Since the computational and infrastructure requirements (training database) during the learning phase are considerably more complex than those for classification, efficiency considerations call for separate hardware for learning and retrieval. Network parameters determined during learning are downloaded into processors specialized for the classification task. This approach, which we focus on here, contrasts with implementations of neural network processors with on-chip learning.[6,7] Those circuits are not suitable for the pattern recognition problems we investigate here, because of limitations of the training algorithms implemented on these chips or because of the limited size of the network that can be trained.

The basic operation performed by a neuron during classi-

fication is a weighted sum, followed by a nonlinear squashing function f, typically a hyperbolic tangent or approximation thereof:

$$y = f(\Sigma_i x_i w_i + b).$$

We generally refer to the inputs x_i of the neuron as connections and the w_i parameters as weights. Each input is either tied to the output y of another neuron or to an external input. Optionally, a bias b may be added to the weighted sum.

The total number of connections in neural networks for applications such as handwritten character recognition may amount to 10,000 to several hundred thousands.[8] Networks that solve more general problems, such as recognition of entire words instead of isolated characters, require even larger numbers of connections. The speed requirements of typical applications call for a few tens to several thousands of classifications per second. For each classification, the network must evaluate one multiplication and one addition for every connection, which translates to a few billion multiply-add operations per second. Only parallel implementations, in which several connections are evaluated concurrently, achieve such computational power.

ANNA evaluates dot products of state and weight vectors and applies a nonlinear squashing function to results.

The most general network topology permits connections between any two neurons. Such a high degree of (possible) connectivity, combined with the need for parallel processing, results in enormous hardware requirements, and therefore calls for a compromise. Usually, the neurons in a network are arranged in layers, each of which receives inputs only from neurons in the previous layer. Layers may be fully connected; that is, each neuron may be connected to every neuron in the preceding layer. Often, however, we use local connectivity to express knowledge about the problem (geometric relations such as the neighborhood of pixels in an image) in the network architecture and thus improve the recognition performance.[1]

For example, the fact that some pixels in an image are adjacent to each other can be built into the network architecture by constraining neurons to receive inputs only from neighboring pixels. In a fully connected topology, such information must be derived from the training set during the learning phase, usually meeting with only partial success.

A neural network processor could be designed to implement only networks with fully connected topology. Local connectivity would then be realized by simply setting the weights of unused connections to zero. Since, in typical neural networks, the ratio of such unused connections to actual connections is easily 100, such an implementation is unacceptably inefficient. The added complexity of the hardware required to support local connectivity is no match for the millions of connections saved.

Another challenge for a compact hardware implementation of a classifier is the amount of memory needed for storing several tens or hundreds of thousands of weights. Fortunately, the weights of many neurons in important connection topologies, including time-delay or feature extraction neural networks,[1,2,9] are identical. In these architectures the connection topology corresponds to a one- or higher dimensional convolution, followed by the nonlinear squashing function, as is illustrated. We can realize such a structure with a single, time-multiplexed neuron with a corresponding saving of storage and computing devices.

We can further optimize the hardware complexity by matching the computational accuracy of the processor to the requirements of typical neural networks. Both experience and theory[10] indicate that neural network classifiers can be designed to be insensitive to low-resolution arithmetic. Experiments with character recognizers show that the recognition performance remains virtually unchanged when the inputs and outputs of the neurons are quantized to 3 bits, and the weights to approximately 5 bits. Higher resolution is required in the last layer for the rejection of ambiguous or unclassifiable patterns. Since in typical neural networks the output layer contains only a small fraction of the total number of connections, we reduce system complexity by evaluating those connections on a different processor with higher accuracy.

ANNA

Figure 1 shows the building blocks of ANNA, a neural network chip that implements the concepts just outlined. It concurrently evaluates several dot products of state and weight vectors and applies a nonlinear squashing function to the results. Data enters the chip through a shift register, which reads up to four values at a time. A file with 16 vector registers stores intermediate results when multilayer networks are evaluated.

The 64-word-wide (3 bits per word) shifter reads up to four inputs in each cycle. In this process, the current shifter contents shift left one to four word positions. The use of a shifter limits the number of pins required and supports convolutional network topologies and multiplexing of neurons with identical weights. The shifter alone handles one-dimensional convolutions, while an external data formatter; for example, a line

delay register,[11] is needed for two- or higher dimensional computations. Data loaded into the chip can be buffered temporarily in a file of 16 vector registers to reduce the required input bandwidth, to evaluate neurons with more than 64 inputs, or to store intermediate results.

Eight banks of vector multipliers perform the actual computation. Each bank consists of a latch to hold the state vector plus eight vector ALUs with 64 synapses each. A multiplexer that can be configured to combine the contributions from one to four vector multipliers connects the outputs from the vector multipliers to the neuron bodies. When the latches of several vector multiplier banks hold different data, the network evaluates neurons with up to 256 inputs. The number of neurons depends on the number of inputs: Extremes of 16 neurons with 256 inputs each, or 256 neurons with 16 inputs, as well as many intermediate arrangements are possible.[12] The topology can be rearranged on a per-instruction basis to permit evaluation of several layers of a network with different architectures on a single chip without performance penalty.

The neuron bodies first scale the output from the vector multipliers by a factor that can be set in the range 1/16 to 1/2 in eight levels to optimize the useful dynamic range of the circuit. Then the neuron bodies evaluate the squashing function and convert the result to the same 3-bit, signed magnitude representation used at the input of the chip. The weights in the vector multipliers are stored as charge packets on capacitors and must be refreshed periodically. Two on-chip digital/analog converters (DACs) update the values of two different synapses in each clock cycle for a refresh speed of 110 μs for the entire array.

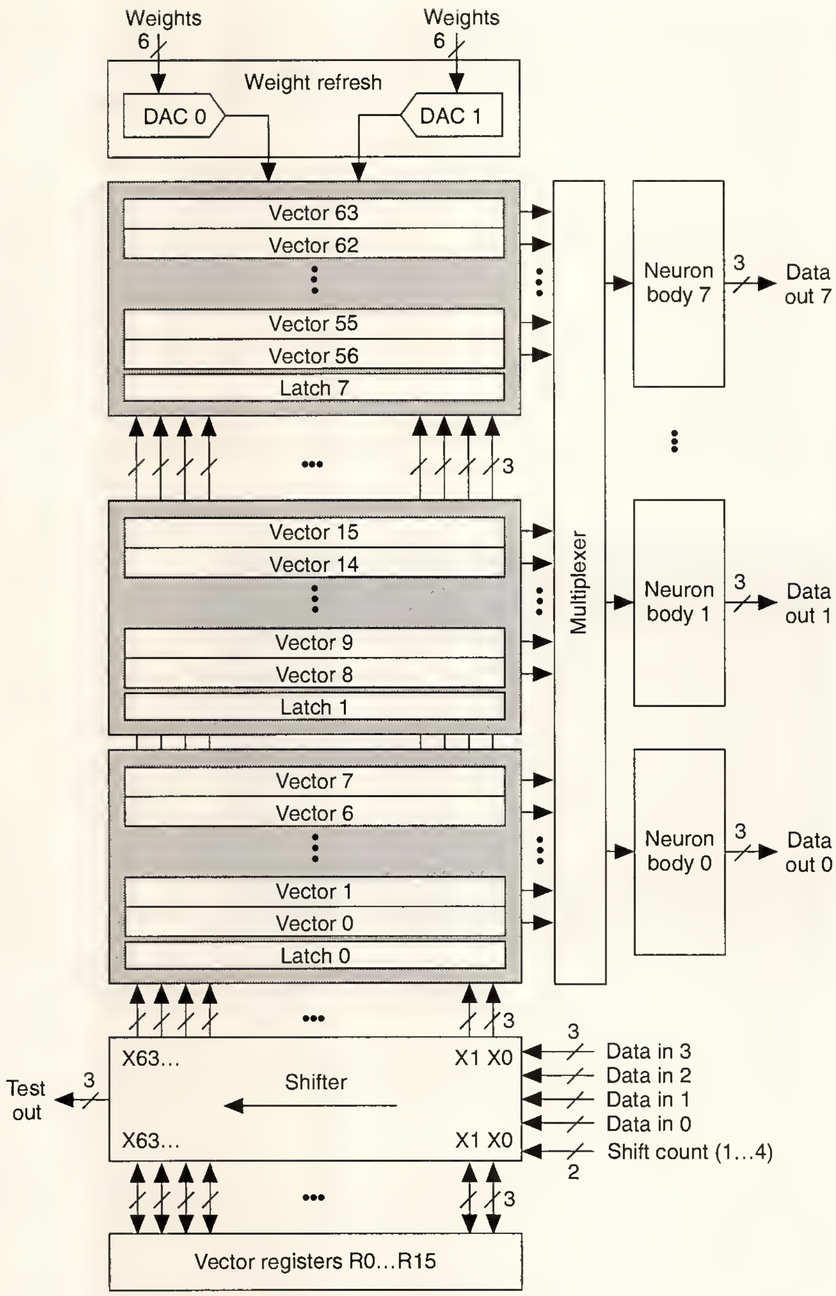


Figure 1. Block diagram of the neural network chip, ANNA.

The chip is programmed with three instructions, CALC, SHIFT, and STORE, to perform computations, load data from an external data source, and transfer data between the shifter, register file, and vector multiplier banks. Parameters for each instruction determine shift count, source of data, number of inputs per neuron, or neuron gain. The CALC instruction executes in four 50-ns cycles, and the network evaluates the other two operations in one clock cycle concurrently with an ongoing CALC instruction. In 200 ns the chip can, for example, load eight states and store them in a register and two latches, and evaluate the dot product and nonlinear function of eight vectors with 256 components each. The weight refresh takes place simultaneously transparent to

Table 1. System features.	
Characteristic	Value
Synapses	4,096
Bias units	256
Synapses per neuron	16 to 256
Weight accuracy	6 bits
State accuracy	3 bits
Input rate	120 Mbps
Output rate	120 Mbps
On-chip data buffers	4.6 Kbits
Computation rate (sustained)	5 Gcps
Refresh (all weights)	110 μs
Clock rate	20 MHz

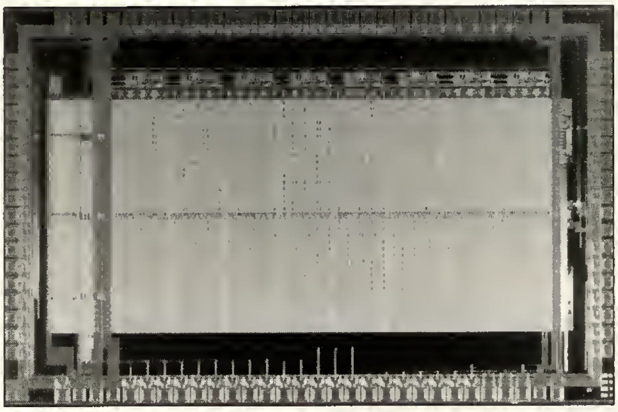


Figure 2. Die photograph. The synapse array can be seen in the center, the shifter and register file on the left, the neuron bodies at the top, and the weight refresh DACs on the right.

the user. Table 1 summarizes the features of the chip.

The chip contains 180,000 transistors and measures 4.5×7 mm^2 (see Figure 2). It was fabricated in single-polysilicon, double-metal, 0.9-μm CMOS technology with a 5V power supply. The current drawn by the chip reaches 250 mA when all weights are programmed to their maximum value but is less than 100 mA in typical operation.

Programmability is one of the key features of the neural network chip. Table 2 lists a selection of network topologies that can be implemented and the achieved perfomance in each case. The chip processes networks with full or sparse connection patterns of selectable size, as well as networks with feedback at a sustained rate of over 10^9 connections per second.

Table 2. Sample network architectures and performance.	
Network topology	Average performance (GC/s)
Fully connected (one layer)	
64 inputs, 64 outputs	2.1
128 inputs, 32 outputs	1.2
32 inputs, 128 outputs	1.2
Local receptive fields	
64 × 1, 64 features	2.3
16 x16, 16 features	4.7
16 × 8, 32 features	3.6
Multilayer network	
64 inputs, 32 hidden,	
32 hidden, 32 outputs	0.8
Hopfield neural network	
64 neurons	2.1

Of particular importance for neural network pattern classifiers are neurons with local receptive fields and weight sharing, such as TDNNs.[9] The neural network chip supports weight sharing in several ways. The shifter and register file enable loading of data and the computation to go on in parallel. Also, data that has been loaded onto the chip once can be buffered and reused in a later computation. Finally, rather than requiring separate hardware for all weights, neurons with identical parameters are stored only once.

Development system

As mentioned before, the characteristics of the ANNA chip—high speed, parallel computation, limited instruction set, and low resolution—differ considerably from those of general-purpose computers. Efficient algorithms that derive optimal benefit from the special processor can be designed only if the processor is available in the early design stages. A development system consisting of an ANNA, a workstation, and appropriate software addresses this requirement. Figures 3 and 4 illustrate the hardware setup, which includes an ANNA and a 20-Mflops DSP32C with 1-Mbyte fast static RAM.

A DMA interface that directly maps the SRAM into the address space of the PC bus or VMEbus exchanges data with the host computer. The DSP, which is also used for pre- and postprocessing and for computations that require higher precision than that of ANNA, generates instructions for ANNA. The entire system is controlled by a program running on the workstation that calls routines and exchanges data with the DSP transparently to the user. The software for the system is written in the high-level language C++, with the exception of a few time-critical routines that are handcoded in DSP assem-

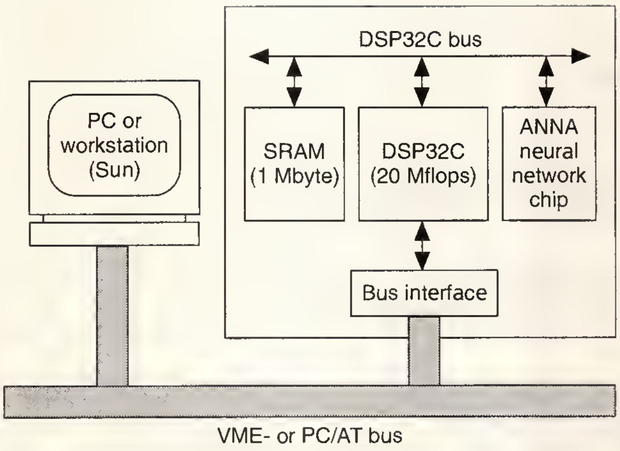


Figure 3. Block diagram of the neural network accelerator board.

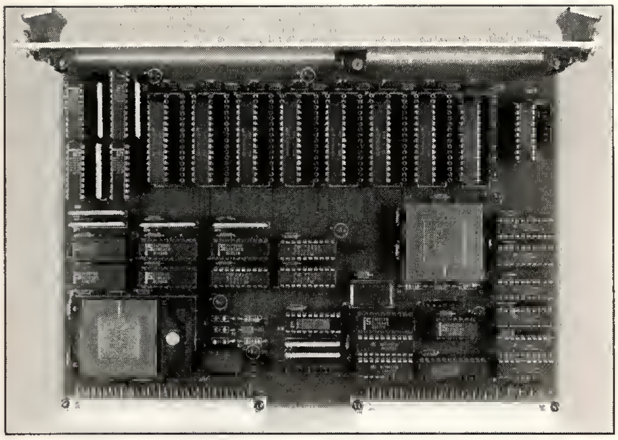


Figure 4. Neural network accelerator with ANNA and the DSP32C.

bly language.

Networks are trained on the workstation or the DSP. The neural network chip can also be included in the training process, for example, to adjust the network to ANNA's low-resolution processing. Training of individual chips is not necessary, however, because of the good matching between individual devices. Once trained, the network topology and weight values are downloaded into the DSP for execution on ANNA.

Character recognition

Speed, capacity, and programmability are important aspects of neural network hardware. Their practical relevance, however, must be proven on a real-world application, such as the implementation of an optical digit recognizer[1] on the neural network chip we describe here. This network has been trained with the backpropagation algorithm[4] to recognize handwritten digits from a 20 × 20-pixel image. The classification error rate on a test set consisting of 2,000 handwritten digits is 4.9 percent miss classifications, compared to a human performance of 2.5 percent on the same data.

Figure 5 illustrates the architecture of the network; Table 3 lists statistical information about each layer. The more than 3,500 neurons with a total of over 133,000 connections are arranged in five layers. The first four layers employ a 2D convolutional topology with various kernel sizes and subsampling factors. Because of weight sharing, the number of weights (free parameters) in these layers is much smaller than the number of connections. The last layer is fully connected. We chose this topology to maximize recognition performance and classification speed of an implementation on a floating-point DSP32C digital signal processor.[13]

Special steps are necessary to adapt the network to the low resolution of the chip. Simple quantization of all weight values

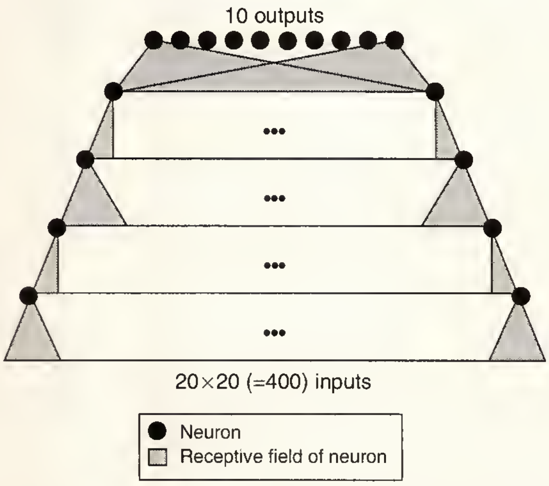


Figure 5. Architecture of the character recognition network.

Table 3. Connectivity of character recognizer neural network.

Layer	Neurons	C/s	Weights
5	10	3,000	3,000
4	300	1,200	12
3	1,200	50,000	500
2	784	3,136	4
1	3,136	78,400	100

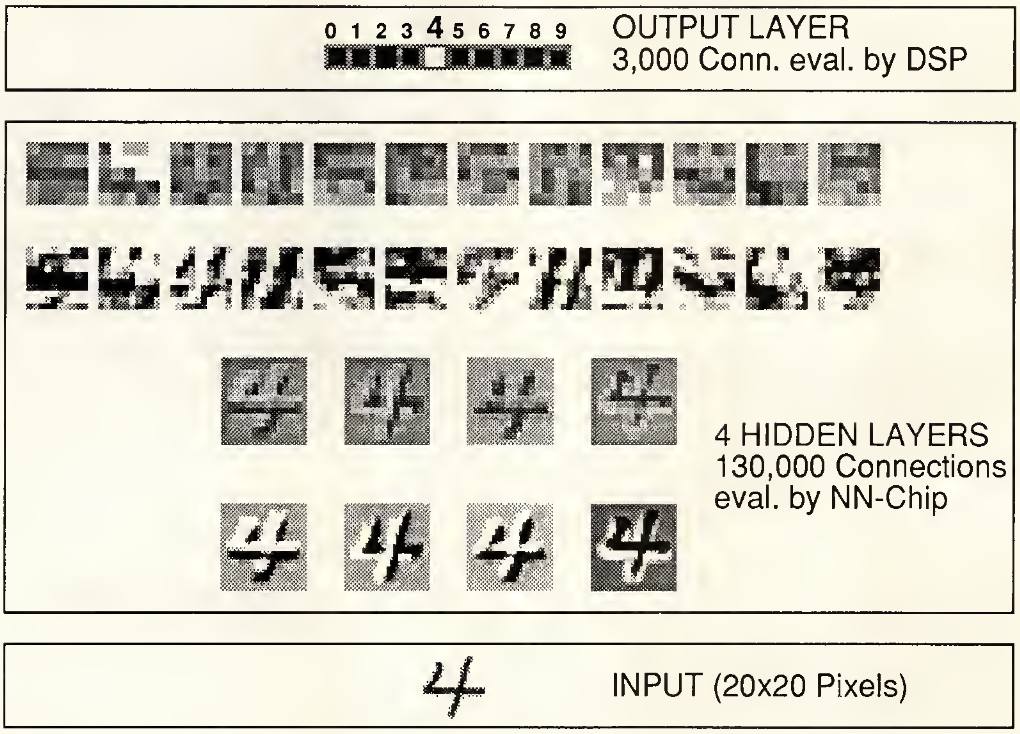


Figure 6. Sample chip output for optical character recognition. The gray levels encode the neuron state.

results in an unacceptable loss of accuracy. However, experiments reveal that the computational accuracy provided by the chip is adequate for all but the 3,000 weights in the last layer of the network. This last layer is retrained with quantized data obtained from the chip to eliminate performance degradation. After retraining, the classification error rate on the test set is 5.3 percent, compared to the original 4.9 percent. This result is obtained consistently with different chips for which the last layer has not been retrained individually. Figure 6 shows the input, output, and internal states of the neural network for a sample input that has been processed by the neural network chip.

The first four layers of the network with 97 percent of the connections but only 616 weights fit on a single neural network chip. The remaining 3,000 connections of the last layer are evaluated on the DSP32C. The throughput of the chip is more than 1,000 characters per second or 130,000 connections per second. This figure is considerably lower than the peak performance of the chip (5G connections per second), a consequence of the small number of inputs of most neurons in the network for which the chip cannot fully exploit its parallelism. Nevertheless, the chip's performance compares favorably to the 20 characters per second that are achieved when the entire network is evaluated on the DSP32C. The recognition rate of the chip is far higher than the throughput of the preprocessor, which relies on conventional hardware. Improvements of both the recognition rate and accuracy can be expected when the network architecture is tuned to take full advantage of the parallelism of the ANNA chip.

NEURAL NETWORKS ARE ATTRACTIVE FOR PATTERN classification applications but suffer in practice from the limited speed that can be achieved with implementations based on classical processors. This problem can be overcome with highly parallel special-purpose VLSI circuits. While a fully parallel implementation of sufficiently large networks is currently not feasible, we can achieve adequately high performance with an architecture that exploits the limited connectivity and weight sharing that are typical for pattern classifiers. We demonstrated this performance with a neural network classifier with over 133,000 connections that has been implemented on a single

neural network chip performing over 1,000 classifications per second. This result eliminates throughput from the constraints faced by network designers. The availability of fast special-purpose hardware for large applications sets the conditions to explore new neural network algorithms and problems of a scale that would not be feasible with conventional processors.

We expect further advances when the architecture of the network is modified to fully take advantage of the chip's parallelism. While the size of the current network has been constrained by the speed of conventional hardware, such issues vanish because of the high speed of the chip. The price for this throughput is the specialization of the circuit, specifically its low resolution, and its focus on neural network algorithms. Future research will benefit from the speed of the novel hardware but must also address questions regarding the limitations of special-purpose hardware. Furthermore, it appears attractive to implement larger tasks (to include image location, segmentation, and scaling into the recognition process) with neural networks to benefit from the powerful hardware.

References

1. Y. leCun et al., "Handwritten Digit Recognition with a Back-Propagation Network," in *Neural Information Processing Systems*, Vol. 2, D.S. Touretzky, ed., Morgan Kaufmann Publishers, San Mateo, Calif., 1990.
2. I. Guyon et al., "Design of a Neural Network Character Recognizer for a Touch Terminal," *Pattern Recognition*, Vol. 24, No. 2, 1991, pp. 105-119.
3. B.E. Boser and E. Sackinger, "An Analog Neural Network Processor with Programmable Network Topology," *ISSCC Dig. Tech. Papers, Proc. IEEE Int'l Solid-State Circuits Conf.*, Vol. 34, Feb. 1991, pp. 184-185.
4. D.E. Rumelhart and J.L. McClelland, *Parallel Distributed Processing—Explorations in the Microstructure of Cognition*, Vol. 1, MIT Press, Cambridge, Mass., 1986.
5. M. Mexard and J.P. Nadal, "Learning in Feed-Forward Layered Networks: The Tiling Algorithm," *J. Phys.*, Vol. A-22, 1989, pp. 2191-2203.
6. Y. Arima et al., "A 336-Neuron, 28K-Synapse, Self-Learning Neural Network Chip with Branch-Neuron Architecture," *ISSCC Dig. Tech. Papers, Proc. IEEE Int'l Solid-State Circuits Conf.*, 1991, pp. 182-183.
7. J. Alspector, R. Allen, and A. Jayakumar, "Relaxation Networks for Large Supervised Learning Problems," *Neural Information Processing Systems*, Vol. 3, R. Lippmann, J. Moody, and D. Touretzky, eds., Morgan Kaufmann Publishers, 1991, pp. 1015-1021.
8. L.D. Jackel et al., "VLSI Implementations of Electronic Neural Networks: An Example in Character Recognition," *Proc. IEEE Int'l Conf. Systems, Man, and Cybernetics*, 1990, pp. 320-322.
9. A. Waibel et al., "Phoneme Recognition Using Time-Delay Neural Networks," *IEEE Trans. Acoustics, Speech, and Signal Processing*, Vol. 37, No. 3, Mar. 1989, pp. 328-339.
10. M. Stevenson, R. Winter, and B. Widrow, "Sensitivity of Feedforward Neural Networks to Weight Errors," *IEEE Trans. Neural Networks*, Vol. 1, No. 1, Mar. 1990, pp. 71-80.
11. P.A. Ruetz, "The Architectures and Design of a 20-MHz Real-Time DSP Chip Set," *IEEE J. Solid-State Circuits*, Vol 24, No. 2, Apr. 1989, pp. 338-348.
12. H.P. Graf and D. Henderson, "A Reconfigurable CMOS Neural Network," *ISSCC Dig. Tech. Papers, Proc. IEEE Int'l Solid-State Circuits Conf.*, Vol. 33, 1990.
13. M.L. Fuccio et al., "The DSP32C: AT&T's Second-Generation Floating-Point Digital Signal Processor," *IEEE Micro*, Vol. 8, No. 6, Dec. 1988, pp. 30-48.

Bernhard E. Boser is an assistant professor in electrical engineering at the University of California, Berkeley. While writing this article, he was working on algorithms and parallel hardware for artificial neural networks at AT&T Bell Laboratories.

Boser received a diploma in electrical engineering from the Swiss Federal Institute of Technology in Zurich, Switzerland, and MS and PhD degrees from Stanford University. He is a member of the IEEE and the Computer Society.

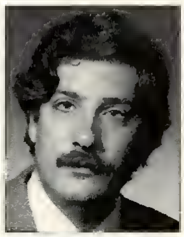
Eduard Sackinger is a member of the technical staff at AT&T Bell Laboratories in Holmdel, New Jersey, on artificial neural-network hardware and its applications to character recognition. He previously worked at the Electronics Laboratory of the Swiss Federal Institute of Technology where he investigated analog applications to floating-gate devices. His research interests include analog circuit design and parallel distributed processing.

Sackinger received the MS and PhD degrees in electrical engineering from the Swiss Federal Institute of Technology (ETH), Zurich, Switzerland. He is a member of the IEEE.

Jane Bromley, a consultant at AT&T Bell Laboratories, works on the application of artificial neural networks to human perception tasks, including the reading of handwritten text. She received the BSc degree in physics and the PhD degree in biophysics from Imperial College, London University. She is a member of the American Psychological Society and the Association for Research in Vision and Opthalmology.

Yann leCun is a member of the technical staff at AT&T Bell Laboratories. He previously worked as a research associate in the Department of Computer Science of the University of Toronto, Canada. His current interests include neural networks and connectionist models, learning theory, pattern recognition, and VLSI implementations of massively parallel systems.

LeCun obtained an engineering diploma in electrical engineering from the School of Electronic and Electrotechnic Engineering, Paris. He holds a PhD degree in computer science from the Pierre and Marie Curie University, Paris, during which time he introduced an early version of the backpropagation algorithm. He served as session chair on the organizing committees of several conferences, including the International Joint Conference on Neural Networks, the Neural Information Processing Systems Conference, and the International Neural Network Conference.



Lawrence D. Jackel heads the Adaptive Systems Research Department at AT&T Bell Laboratories in Holmdel, New Jersey, where he pursues research into theory, applications, and hardware for machine learning and machine perception. His initial research was in microfabrication and microscience.

Jackel received a BS degree in physics from Brandeis University in Waltham, Massachusetts. He was awarded the PhD degree in experimental physics from Cornell University in Ithaca, New York. He is a member of the IEEE, the Computer Society, and the American Physical Society.

Direct questions concerning this article to Bernhard E. Boser, EECS Dept., Cory Hall, University of California, Berkeley, CA 94702.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

High 156 Medium 157 Low 158

IEEE MICRO

Editorial Calendar

APRIL 1992

Hot Chips III

- This extremely popular issue presents the latest developments in microprocessor and chip technology used to construct high-performance workstations and systems as presented at the annual IEEE Computer Society-sponsored symposium
- Past contributors include: Intel, Motorola, Apple, Sun, IBM Research, Metaflow Technologies, Intergraph, ITT

Ad closing date: March 1

JUNE 1992

Associative memories and processors

- Late-breaking developments in wide-word content-addressed memories (CAMs)
- Dynamic CAPP (parallel processor) architectures
- Fault-tolerant architectures for crucial control systems such as those in trains, space systems, etc.

Ad closing date: May 1

AUGUST 1992

European industry

- Recent developments in integrated circuit and microsystem technology from major European manufacturers

Ad closing date: July 1

OCTOBER 1992

Processing hardware for video communication

- ICs for HDTV compression
- ICs for HDTV communication
- Parallel processing systems with real-time video compression capabilities
- Multimedia systems with real-time video compression capabilities

Ad closing date: September 1

DECEMBER 1992

Special Signal Processors

- Recent information from the vital area of digital signal processors
- News about other techniques for signal processing
- Mixed analog/digital processors solve a real need
- Neural networks process signals following methods used by the human brain

Ad closing date: November 1

FEBRUARY 1993

Automotive/traffic microelectronics

- Worldwide developments in microelectronics for traffic and driving assistance
- Latest developments in the European Prometheus and US IVHS programs
- News from Japan, too
- Improving traffic safety with electronics

Ad closing date: January 2

IEEE Micro helps designers and users of microprocessor and microcomputer systems explore the latest technologies to achieve business and research objectives. Feature articles in *IEEE Micro* reflect original works relating to the design, performance, or application of microprocessors and microcomputers. All manuscripts are subject to a peer-review process consistent with professional-level technical publications. *IEEE Micro* is a bimonthly publication of the IEEE Computer Society.

Advertising information: Contact Marian Tibayan, Advertising Department, IEEE Computer Society, PO Box 3014, Los Alamitos, CA 90720-1264; (714) 821-8380; fax (714) 821-4010.

Articles may change. Please contact editor to confirm.

Experimentation with Hypercube Database Engines

Effective algorithms appropriately selected for each task are essential if we are to take full advantage of parallelism in database systems. Some algorithms perform faster than others, depending on the uniqueness value of the data and the uniformity of distribution among the nodes in the system. A performance evaluation under varying experimental parameters determines the optimality of a given algorithm for a particular database task.

Ophir Frieder

Vijaykumar A. Topkar

Ramesh K. Karne

Arun K. Sood

George Mason University

Interest in multiprocessors for database processing has grown beyond the research communities, into the development sectors. As the volume and variety of data stored, accessed, and manipulated grows, computer and communications designers are focusing on parallelism–in particular using general-purpose commercial multicomputers–as a way to meet database processing needs.[1-3]

Using one such computer, Intel's iPSC/2 hypercube, we measured the relationship between packet size, method of clustering messages, and internode traffic on the total sustained communication bandwidth. Having measured the costs associated with internode communication, we then analyzed duplicate removal algorithms. Duplicate removal is an integral part of several frequently used relational database operations, including PROJECT and UNION. We believe it is, therefore, important to develop efficient algorithms.

We also studied the effects of nonuniformly distributed attribute values and tuples across processors on three proposed duplicate removal algorithms. We chose algorithms to represent the several available in the literature.[4-7] We then evaluated the output collection time.

A tutorial on multiprocessor/multicomputing databases[6] or current generation multicomputing architectures[8] is beyond the scope of this article. We intend only to present a brief overview of the iPSC/2's hypercube message-passing system, and discuss the results of our experimentation and analysis. Although our algorithms were implemented on the iPSC/2, we believe they would work as well on other hypercube machines, such as Ncube[9] and Floating Point Systems' T series.[10]

iPSC/2

Intel's Personal Super Computer, the iPSC/2, is a multiple-instruction, multiple-data (MIMD), multicomputer with nodes connected via a hypercube topology. Although many definitions for a multicomputer system exist, we define it as a system comprised of multiple nodes, each of which is a complete computer. That is, each node has a set of resources (dedicated I/O, memory, and CPU) and is under the independent control

of its own operating system.

Message-passing application interface. Application programs access the iPSC/2's message-passing system through system calls.[11-13] The NX/2 operating system provides three levels of system calls: synchronous, asynchronous, and interrupt-driven. Users can mix and match these three types of calls. For example, a message sent with an asynchronous call can be received either with a synchronous call or an interrupt-driven call. The size of the message is limited only by the amount of physical memory available at the node.

The processor uses an application-development message-passing paradigm. Consider, for example, a node program that receives data from the host or from another node and then locally processes them. If the node cannot locally process without the received data, then it must use a synchronous receive call to receive the data.

On the other hand, if the node can accomplish some of the local processing without the received data, then it uses an asynchronous receive. After initiating the asynchronous receive, the node carries out the local processing that does not require the received data. The node program then checks if the data have been received. If the data have not been received, the program waits.

The synchronous calls are also called blocking calls. When a node sends a message using this protocol, it is blocked for processing until the operating system copies the send buffer into its local buffer and is ready to send this message to its destination. This operation does not imply however that the message sent is received at the destination node. The message buffering and flow control in NX/2 ensure the synchronous transmission intended by the application.

The asynchronous calls are nonblocking calls. When a node sends a message using this protocol, the sending node is free to process soon after the execution of that call. The receiving node must poll for any messages received asynchronously. This protocol facilitates the application to perform other tasks while the messages are in transit.

Message-passing measurements. We conducted the following experiments to analyze the communication behavior of iPSC/2's message-passing system. We measured the communication times between two adjacent nodes that are also at the extreme diagonals on the hypercube. The communication times for a one-hop (adjacent nodes) and multihop (nonadjacent nodes) are almost equal, because the intermediate nodes take only a few microseconds to help establish a path. Thus, their computation phase is uninterrupted. This indicates that the iPSC/2 message-passing interconnection, though organized as a hypercube topology, practically functions as a fully connected interconnection network.

A fully connected 128-node system has eight simultaneously available paths, that is, paths without contention. However, If multiple nodes simultaneously send messages to a common destination, then contention is imminent. To evaluate the effects of contention we measured the communication times when one or more source nodes send messages to a single receiving node. The communication times at the source nodes slow down proportionally to the number of source nodes increases.

In our experiments, we used an eight-node system and measured average communication time of 1,000 messages in which each node sends to and receives from another node. Figure 1 shows the communication time for varying message

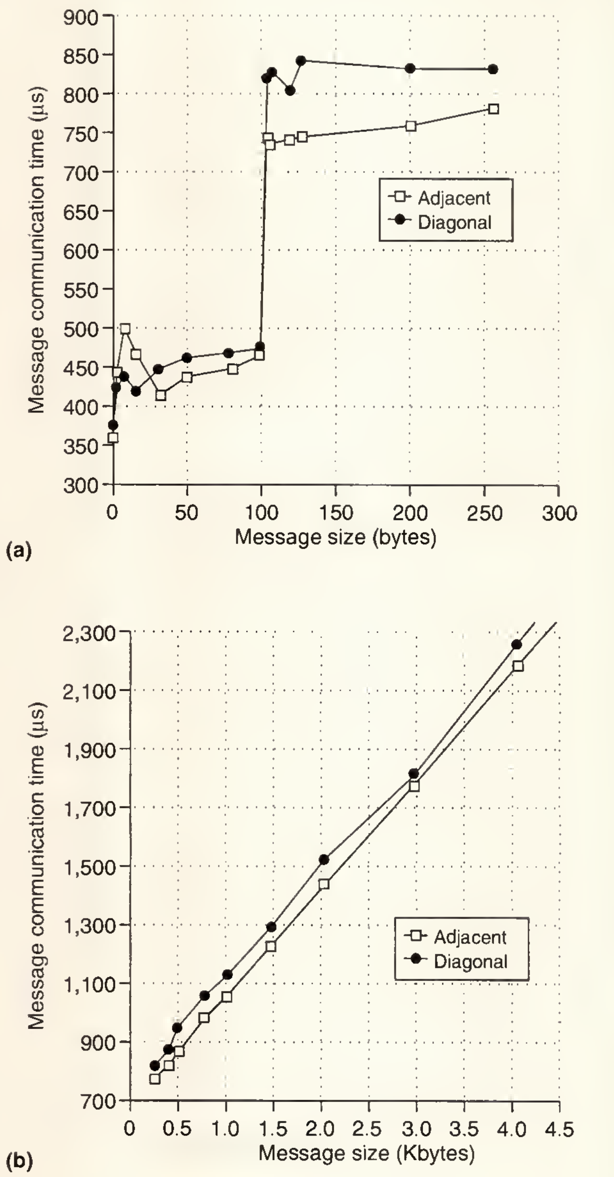
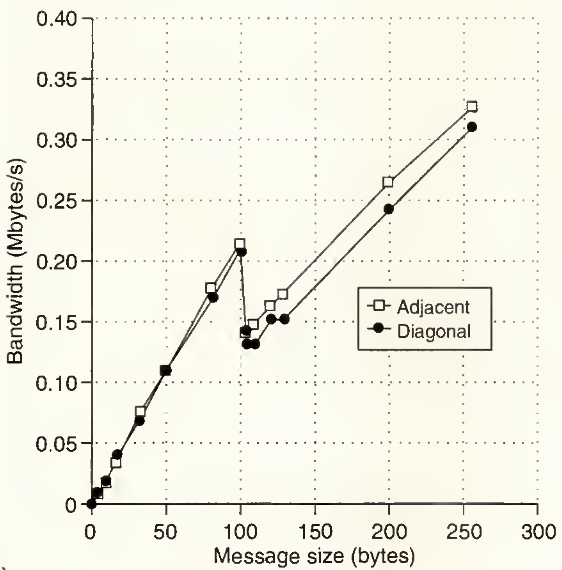
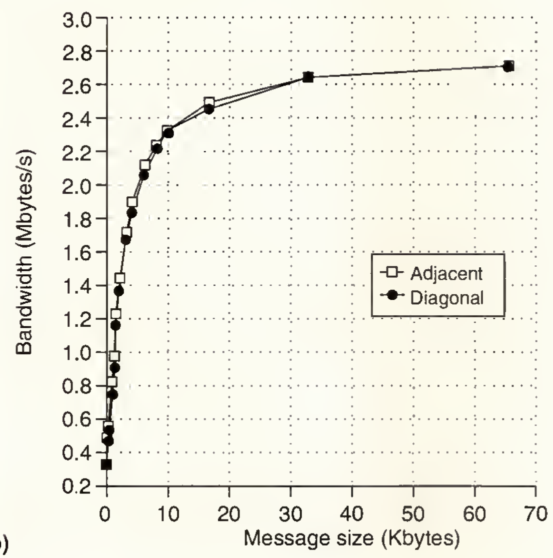


Figure 1. Communication time as a function of message size: small (a), large (b).



(a)



(b)

Figure 2. Communication bandwidth as a function of message size: small (a), large (b).

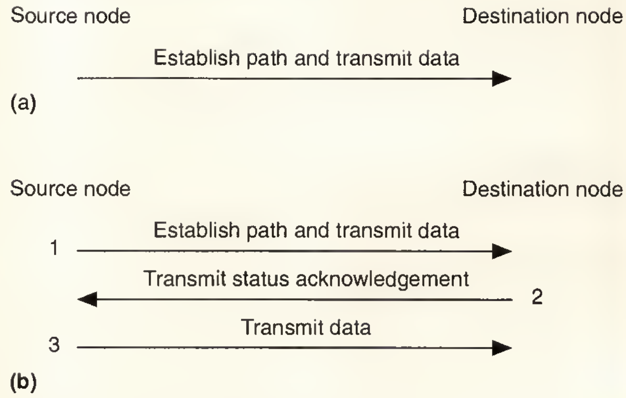


Figure 3. Message transmission protocols: one-step (a), three-step (b).

sizes between neighbor nodes and nodes three hops apart. Figure 2 illustrates the computed bandwidths of the communication channel for different operating conditions. Note that the communication time for longer messages is comparable to that of smaller messages. Consequently, we prefer implementations requiring fewer but longer messages.

We measured the communication time for a zero-byte message at 361 µs. This is the minimum setup time required to transmit a message of any size. In addition to the setup time, messages larger than zero bytes need a propagation time.

The system uses two protocols for message transmission. For short messages (zero to 100 bytes), a one-step protocol, shown in Figure 3a, sets up a path and transmits data concurrently.

Figure 1a shows that, for messages larger than 100 bytes, the communication time increases by almost 280 µs. (Since all messages include a 4-byte boundary, a 100-byte message actually takes 104 bytes.) The time increases because the system uses a three-step protocol, shown in Figure 3b, for larger messages. In the first cycle, the source node establishes a path by sending a status message to the destination node. In the second, if the destination node is ready to receive a large message, it sends back a status acknowledgment. In the final step, the source node sends the complete message to the destination node.

This three-step protocol increases the communication time for large messages, but two different protocols are required since the one-step protocol could overflow the memory buffer if used for large messages. Alternatively, the three-step method would add unnecessary protocol overhead if used for small messages.

Figure 1b shows the communication time chart for messages between 256 and 4,096 bytes. Note that communication time for extreme diagonal nodes and adjacent nodes is almost equal. Figure 2a shows the bandwidth chart for messages from 0 to 256 bytes. Figure 2b shows the bandwidth chart for messages in the range from 256 bytes through 64 Kbytes. To achieve a 2.7-Mbyte/s bandwidth, the message size must be at least 64 Kbytes. For small messages, say 100 bytes, the bandwidth is 0.21 Mbytes/s.

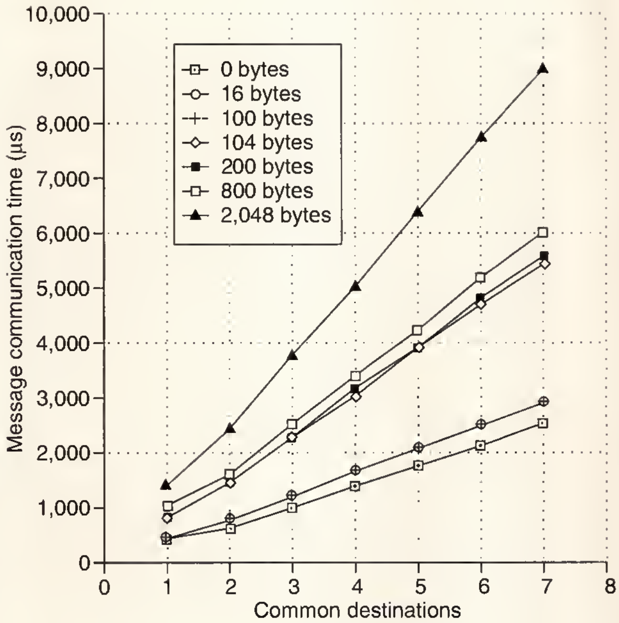


Figure 4. Number of common destinations versus time.

We also measured communication times for message transmissions with contention in the network. Contention is created by transmitting messages from one or more source nodes to a common destination node. To demonstrate, we chose an eight-node hypercube with an arbitrarily chosen destination node 5. Nodes 0 through 4, 6, and 7 transmit messages to node 5 and the transmission times are recorded.

Figure 4 shows the number of common destinations versus message communication time. Each curve represents a different message size. The message communication time increases almost linearly, as the number of common destinations increase. For example, sending a 100-byte message from a single source node takes 501 µs, while for seven source nodes, the communication time is 3,006 µs. For every 100 bytes of memory, the communication time increases approximately 400 µs for each common destination node added. Similarly, for a 104-byte message, the communication time increases approximately 800 µs for each common destination node. (The dramatic increase for a 104-byte message is due to the three-step protocol.)

We also observed that the communication time increase caused by contention is independent of the location of the common destination node. This increase stems from the communications technology employed in the iPSC/2. These measurements suggest that, to achieve scalable performance, we should use algorithms that avoid network contention.

Our experiment led us to the following conclusions:

- *Designers should choose an appropriate message size based on the requirements of an application.* The three-step protocol causes extra overhead in the transmission of messages. Thus, if the message size is only slightly over 100 bytes, it may be advantageous to reduce the message size.
- *For short message transmissions, the iPSC/2 offers a quite low bandwidth, on the order of Kbytes.* Thus, message clustering may be required to achieve higher performance. In message clustering, one or more messages destined for the same target node are collected in a packet and transmitted as a single message packet. However, message clustering increases processing overhead. For some applications, this increased overhead may nullify the benefits of using the high bandwidth.
- *The iPSC/2 effectively supports a fully connected network.* With the notable exception of minimizing node contention, there are few additional benefits to partitioning the application to suit the adjacent node communications.
- *Simultaneous data transfer to the same node dramatically increases the communication time.* Consequently, we prefer algorithms that avoid such contention.

We designed the algorithms presented in the following sections with these communication constraints in mind.

Relational database

Numerous commercial relational database systems are available that execute on a range of host processing systems including personal, mini- and mainframe computers, as well as a host of multicomputer environments. Each system's performance depends on the underlying execution environment and the liberties taken by the software developer. For example, duplicate tuples (or rows) are not allowed in the actual relational database model,[14] and yet several vendors support duplicate tuples. We initially describe a subset of relational operators and then provide a discussion of three duplicate removal algorithms that can be employed as part of the implementation of these operators.

Operators. The relational database model is the underlying environment for a wide diversity of applications. For example, medical pictorial databases, commonly called picture archiving and communication systems (PACS), employ the relational model to store and access patient data.[15] Some designers propose protocol verification systems that use the relational model to implement an efficient algorithm based on a reachability analysis technique.[16] Thus, the development of high-performance parallel algorithms for the relational operators impacts not only the traditional consumer database processing arena but also nontraditional database domains, including the communications and medical communities.

Table 1. Relations r and s.

Relation *r*		Relation *s*		
Employee number	Age	Employee number	Children	Gender
0	33	0	3	M
1	25	1	0	M
2	53	5	3	F
3	25	7	2	F
4	25	8	1	F
5	45	10	4	M
6	28	16	8	F
7	45	17	4	F
8	64	18	2	M
		22	1	F
		28	0	M
		31	0	F
		42	3	F
		43	2	M

To review relational database nomenclature, consider the relational database shown in Table 1. This database consists of two relations, *r* and *s*. Relation *r* has two attributes: employee number and age. Relation *s* has three attributes: employee number, number of children, and gender.

Each attribute is defined over some finite or countably infinite domain. In this example, the attributes in *r* are defined over the set of whole numbers and the set of whole numbers between 0 and 120. The attributes in *s* are defined over the domains of the set of whole numbers, the set of whole numbers between 0 and 30, and the set {M, F}. Relation *r* consists of nine tuples; each tuple contains two attributes:

{<0, 33>, <1, 25>, <2, 64>, <3, 25>, <4, 25>, <5, 53>, <6, 28>, <7, 45>, <8, 64>}.

Similarly, *s* consists of 14 tuples, each comprised of three attributes.

Several popular operators exist for the manipulation of the relations comprising the database. Here, we briefly overview only four: Project, Union, Select, and Join. Formally specified, these operators are defined as follows.

- *Project.* The projection on $P[XYZ]$, denoted as $\pi_A(P)$, is defined by $\pi_A(P) = \{ x[A] \mid x \in P \}$, where A is a set of attributes of P.

- *Union.* The union of two relations, $P[XYZ]$ and $Q[XYZ]$, denoted as $P[XYZ] \cup Q[XYZ]$, is defined by $P[XYZ] \cup Q[XYZ] = \{ x \mid x \in P \text{ or } x \in Q \}$, where X, Y, and Z are a disjoint set of attributes.

- *Select.* The selection on $P[XYZ]$, denoted as $\sigma_{B \oplus b}(P)$ where $\oplus \in \{<, \leq, =, \geq, >, \neq\}$ is defined by $\sigma_{B \oplus b}(P) = \{ x \mid x[B] = b, x \in P \}$, where B is an attribute of P.

- *Join.* The join of two relations $P[XYZ]$ and $Q[VWX]$, denoted as $P[XYZ] \mid x \mid Q[VWX]$, is defined by $P[XYZ] \mid x \mid Q[VWX] = \{ u \mid p \in P, q \in Q, p[X] = q[X], u[XYZ] = p, u[VWX] = q \}$, where V, W, X, Y, and Z are a disjoint set of attributes. If no common joining attributes exist, the join of P and Q is the Cartesian product of P and Q.

Given relations *r* and *s* as shown in Table 1, the user requests, or queries, are evaluated in Figure 5.

In typical database applications, most attributes contain duplicate values. Duplicates occur frequently in database processing due to the inherent nature of the data being processed. For example, George Mason University has about 20,000 students in 43 undergraduate programs. Therefore, the program discipline attribute in the student database contains a high degree of duplicates, that is, a low uniqueness factor.

1. Find all employees with three children.

   $\sigma_{\text{Children}=3}(s) =$

Employee number	Children	Gender
0	3	M
5	3	F
42	3	F

2. Find all employees that are childless or are 25 years old.

   $\pi_{\text{Emp No.}}(\sigma_{\text{Children}=0}(s)) \cup \pi_{\text{Emp No.}}(\sigma_{\text{Age}=25}(r))$

Employee Number
1
3
4
28
31

3. Determine all the available information about 45-year-old women.

   $\sigma_{\text{Age}=45}(r) \mid x \mid \sigma_{\text{Gender}=F}(s)$

Employee number	Age	Children	Gender
5	45	3	F
7	45	2	F

Figure 5. Queries.

```
If an attribute value in the local database is also present in the received packet, then
{
    if (local_count > packet_count) then
        local_global_count = local_global_count + packet_count
    else if (local_count < packet_count) then
        delete the tuple from the local database
    else if (local_count = packet_count) then
    {
        if (local_node_address > packet_node_address) then
            local_global_count = local_global_count + packet_count
        else
            delete the tuple from the local database
    }
}
```

Figure 6. Multiple-message ring algorithm (local processing).

In the database presented in Figure 5, duplicate entries appear in the number of children and employee gender attributes of *s*, and in the age of the employees attribute in *r*. In fact, in most cases, only the key attributes are typically unique. We focused on designing algorithms that remove duplicates.

We also studied the effects of low uniqueness on the performance of proposed parallel algorithms. Other authors have written on the computational complexity and processing requirements of relational database queries in the presence of duplicate attribute values.[7,17,18]

Of the relational database operators presented earlier, we emphasize Project and Union since both can result in duplicate tuple values in the initial stage of processing. That is, the Project operator initially eliminates columns which may result in duplicate tuple values. Thus the removal of duplicates from the generated set of tuples must follow.

The Union operator eliminates the duplicate copies that result from the initial processing stage. In the Union operator, the initial processing stage involves the combining of the tuples of the first relation with the tuples of the second relation. Duplicate removal comprises the second stage. (In a distributed-memory architecture, the first stage can occur without any internode communication, whereas, in the second stage, all nodes must communicate.) The duplicate tuples belong to the intersection of the input relations to the Union operation.

Algorithms for duplicate removal. We describe three parallel duplicate removal algorithms, the multiple message ring (or, simply, the ring), recursive reduction, and bucket algorithms. Each algorithm removes the duplicates in two steps: locally at each node, and then globally throughout the hypercube.

All three algorithms commence with the local duplicate removal phase. This phase forms a local database of tuples with unique attribute values originally available at the given node (that is, no two tuples have the same attribute value). For each unique datum value, two fields are maintained. One is a local repetition count representing the number of copies of the datum initially present at the node. The other is a global repetition count that is initialized to the corresponding local repetition count before the global duplicate removal stage.

Since the local duplicate removal phase is identical in all three algorithms, we will not elaborate on it. We will briefly describe the global duplicate removal phase for each algorithm, though further details of the algorithms, including analytical models and performance evaluation, are beyond the scope of this article.[7]

Multiple-message ring. After the local duplicate removal phase, a unidirectional ring is embedded using Reflexive Gray Codes in the $\log_2 N$-dimensional hypercube. A packet consisting of tuples with locally unique attribute values and their local repetition counts is formed at each node. The global duplicate removal is achieved in $N-1$ iterations. During each iteration, each node sends a packet to the next node, in a pipeline ordering of the Reflexive Gray Code, and receives one from the previous node. The received packet updates the local database as shown in Figure 6. This procedure repeats for $N-1$ iterations, after which the following are true:

- Attribute values are unique across node boundaries.
- The node on which an attribute value is found is the node which had the largest local repetition count to start with. If more than one node had the largest local repetition count to start with, then the node with the largest node address will retain that unique value.
- The final local_global_count of an item is the global repetition count of that value.

Recursive reduction. The global duplicate removal phase is achieved by a $\log_2 N$-fold recursive reduction of the size of the cube containing the data, with the final results placed at a single target node. In each step j, the size of the active cube is halved based on the value of the jth bit in the node address. That is, all nodes with the jth bit equal to one belong to the first half while the remaining nodes, those nodes with the jth bit equal to zero, belong to the second half.

For each node in one half, a corresponding node exists in the other half, where every bit except j is identical in value. Nodes that differ from the target node in the jth bit transmit local data to their corresponding node. The receiving nodes

```
If a datum in the packet is also present in the local database then
        local_global_count = local_global_count + packet_count
    else
        add the datum value to the local database at the appropriate location.
```

Figure 7. Recursive reduction algorithm (local processing).

```
Let N = Number of nodes, k = log2N, Node address = <<n(k-1) n(k-2) ... n0 >>
Let max_value and min_value define the range of attribute values to be sorted.
Let recursion_no = 0;
while (recursion_no < k)
{
  create lower bucket consisting of attribute values such that
      attribute value ≤ (Maximum attribute value – Minimum attribute value) / 2
  create upper bucket consisting of remaining attribute values.
  Let X = <<00 ... 010 ... 0>> where the 1 is in the (recursion_no)th position.
  Let partner_node = <<n(k-1) n(k-2) ... n0>> BIT_WISE_EX_OR X
  if (n(recursion_no) = 0)
  {
      send upper bucket to the partner_node
      receive bucket from partner_node and merge it with the lower bucket
      max_value = min_value + (max_value – min_value) / 2
  }
  else
  {
      send lower bucket to the partner_node
      receive bucket from partner_node and merge it with the upper bucket
      min_value = min_value + (max_value – min_value) / 2
  }
  recursion_no = recursion_no +1
}
```

Figure 8. Pseudocode of bucket algorithm.

merge their respective local databases with the packets received, as shown in Figure 7. On termination, all the unique values and their global counts reside in the target node.

Bucket. This algorithm is a special case of the hash join algorithm[19] and can be used whenever the range of attribute values in the system is known a priori or can be readily and efficiently computed. A bucket consists of attribute values within a specified range. Each node is mapped to a bucket, and the attribute values residing on all other nodes belonging to that bucket are routed to that node. The routing is done in $\log_2 N$ steps by following an algorithm similar to the recursive reduction algorithm. During each step j, corresponding nodes partition the attribute values range into two halves. Nodes whose jth address bit equals zero keep all tuples whose attribute values map to the lower attribute value range and transit the remaining tuples to their corresponding node. Nodes whose jth address bit equals one keep all tuples whose attribute values map to the upper attribute value range and transmit the remaining tuples to their corresponding node. Figure 8 outlines the pseudocode description of the bucket algorithm.

Communication requirements. We designed these three algorithms to exploit our knowledge of the iPSC/2's message-passing system and to prevent simultaneous transmission to a node. In general, the internode communication needs of the algorithms are restricted to adjacent nodes.

Unfortunately, the volume of data transferred between nodes is data dependent. Hence, it is difficult to maximize the sustained communication bandwidth between nodes by using large packets. When multiple packets are required to transfer the data between the nodes however, we prefer fewer large packets to many small packets.

Performance evaluation

The time required for duplicate removal indicates the efficiency of the algorithm. The total time consists of three components:

- local duplicate removal,
- global duplicate removal, and
- data collection.

The ring and bucket algorithms distribute the resulting data across all the nodes. If the results must reside at a single node, they must be transmitted. The time needed to accomplish this is called collection.

The inclusion of collection time and the duplicate removal phase, and the proper selection of the algorithm is application dependent. For example, if the starting data are already locally unique, the local duplicate processing time need not be included. Similarly, if the final results are to be used as an input for further processing, the collection time should not be included in the execution time. Unless otherwise stated, we assume that the local duplicate removal time is included

and the collection time is not included for computing the total execution time of an algorithm.

We carried out experiments by varying the work load characteristics and the system configuration, namely the number of nodes. The cardinality, uniqueness, tuple size, distribution of the unique values, and distribution of data among the nodes define the work load characteristics. We define the uniqueness factor (or simply the uniqueness) of the data as the number of distinct attribute values expressed in a fraction of the total number of attribute values. In our experiments the attribute value of a tuple is taken as an integer. We refer to this integer value as the data value.

As an example, a typical run may consist of eight nodes, 8,000 data values, and a uniqueness of 0.4. The database randomly generates the required data and then distributes the data among the nodes, either uniformly (each node receives the same number of data values) or nonuniformly.

We used each of the three algorithms for parallel duplicate removal and recorded the execution times. The execution times may include the local duplicate removal time and the collection time, as described earlier. Since the data are generated randomly, we expect the execution time of an algorithm to change every time the experiment is carried out. To account for the random behavior, we conducted the experiments multiple times and observed the deviation of the execution times of an algorithm.

We found a small deviation in the execution times for all the runs except those with nonuniformly distributed data among the nodes. Therefore, for the data distributed uniformly among the nodes, we took the average of five runs as the true execution time of an algorithm. For the nonuniformly distributed data, we took the average of 25 runs of each algorithm as the execution time.

Our objective was to evaluate the performance of each algorithm under different work load and system conditions with the eventual goal of dynamically selecting the algorithm of choice for each application. The independent variables were

- the number of nodes,
- the number of data elements,
- the size of each element (or tuple),
- the uniqueness of the data (degree of skew),
- the initial distribution of the data, and
- the desired final distribution of the data (distributed or centralized).

Note that the choice of the optimum algorithm must be determined not only by the performance evaluation but also by the constraints on the initial data and final results.

Figure 9a shows the execution times (including the local duplicate removal times) of the three algorithms and an optimal algorithm as a function of the uniqueness. The system consisted of 16 nodes and 16,000 data values. The optimal

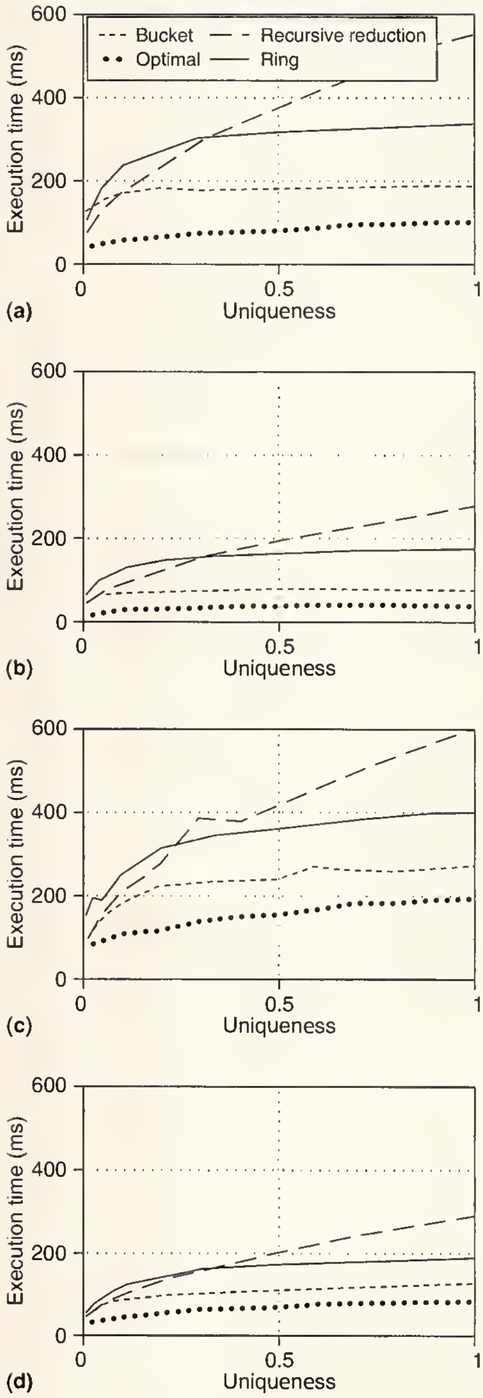


Figure 9. Execution times including the local duplicate removal times: 16 nodes, 16,000 data values (a); 16 nodes, 8,000 data values (b); 8 nodes, 16,000 data values (c); and 8 nodes, 8,000 data values (d).

algorithm using N nodes requires $(1/N)$th the time taken by an optimal uniprocessor algorithm (a tree duplicate removal algorithm, in our case).

Note that for the bucket algorithm the speedup in the execution time is about $N/2$ when N nodes are used. Moreover, the speedup is the same for all values of uniqueness. The speedups for the ring and recursive reduction algorithms are small for high uniqueness values. But the speedups increase as the uniqueness decreases.

Figures 9b, 9c, and 9d show similar plots for 16 nodes, 8,000 data values; 8 nodes, 16,000 data values; and 8 nodes, 8,000 data values. The bucket algorithm is insensitive to the uniqueness of the data, except for very low uniqueness values. The recursive reduction is the most sensitive, and the ring algorithm is moderately sensitive to the uniqueness.

The bucket algorithm performs fastest for all except the very low values of uniqueness. For completely unique data (a uniqueness factor of 100 percent), the recursive reduction algorithm takes almost twice as long as the ring algorithm. But the performance of the recursive reduction significantly improves as the uniqueness decreases, due to its sensitivity to the uniqueness of data. At lower levels of uniqueness, the recursive reduction actually performs better than the ring. This threshold value of the uniqueness changes with the system and data conditions, namely the number of nodes and the number of data values. For example, for 16 nodes and 16,000 data values, the threshold uniqueness is 0.28, whereas for 16 nodes and 8,000 data values, it is 0.31.

Three components of the execution time explain sensitivity towards uniqueness.

- **Local duplicate removal.** Consider a database of size M with a global uniqueness p. Divide the database into N parts and let the local uniqueness of each part be q. We see that, in general, q will be larger than p. In fact, for $p >> 1/N$, $q \approx 1$. For example, for $M = 16{,}000$ and $N = 16$, if p is reduced from 1.0 to 0.1, q remains at 1.0. Consequently, the local duplicate removal time is not sensitive to the global uniqueness p, especially for large values of p.[7]
- **Global duplicate removal.** This component consists of two parts:
  Data transfer. In the bucket and ring algorithms, the local uniqueness factors determine the size of packets to be transferred. For both algorithms, the packet size is not affected by change in the global uniqueness p. On the other hand, in recursive reduction, the size of the cube decreases as the recursion progresses and the local uniqueness changes from 1.0 (in the first recursion) to p (in the last recursion step). As a result, the size of the packets to be transferred increases from the first to the last recursion and is sensitive to the global uniqueness.
  Local processing. For the bucket algorithm, the size of the received packet and the size of the local database are unaffected by the change in the global uniqueness. As a result, the local processing time of the bucket algorithm is insensitive to global uniqueness. For the ring algorithm, the size of the received packets and the initial size of the local database do not depend upon the global uniqueness. As the iterations progress however, the local database shrinks. The rate of this decrease depends on the global uniqueness. The lower the global uniqueness, the faster the local database shrinks. Hence, the local processing time is moderately sensitive to the change in the global uniqueness. For the recursive reduction algorithm, as explained earlier, both the size of the received packets and the size of the local database are sensitive to the global uniqueness. Consequently, the local processing time of the recursive reduction is highly sensitive to the global uniqueness.

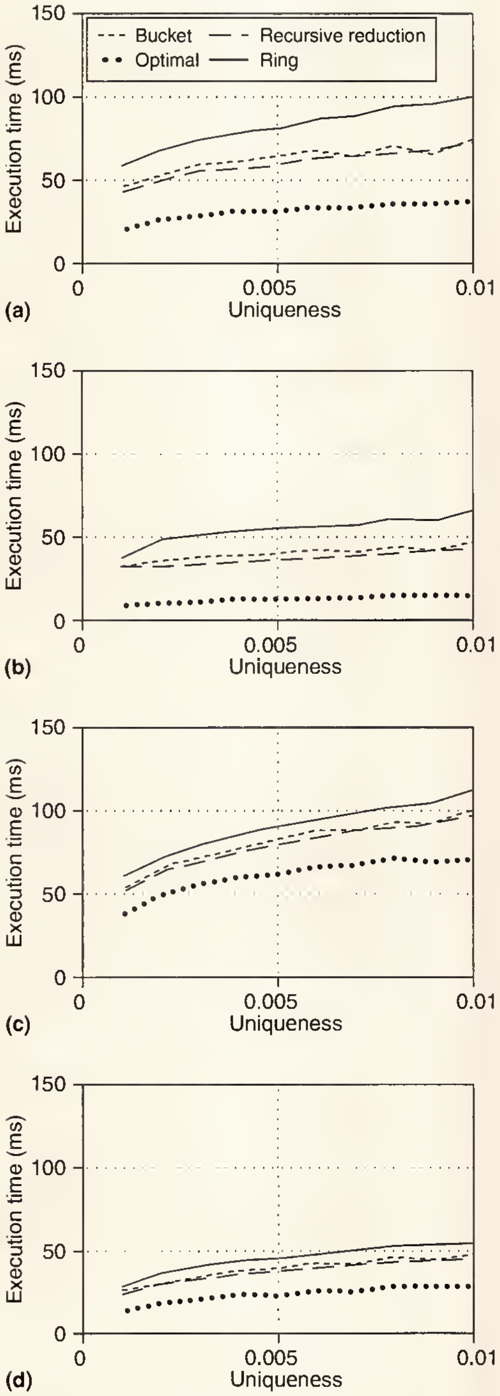
Depending on the particular application at hand, the local duplicate removal time may or may not be important. If this time is not taken into account, the execution times of the three algorithms decrease by a fixed amount. Since the local duplicate removal time changes with uniqueness, the decrease in the execution time–though exactly the same for all three algorithms–differs for each uniqueness. Even if we do not consider the local duplicate removal time, the threshold value of the uniqueness, below which the recursive reduction algorithm performs better than the ring algorithm, does not change.[7]

The performance of the three algorithms for very low uniqueness values, shown in Figure 9, is of interest. Figure 10 shows execution times of the three algorithms for uniqueness varying from 0.001 to 0.01. Note that the ring algorithm is the most time-intensive of the three. The recursive reduction performs better than the bucket for most uniqueness values.

In the ring and the bucket algorithms, the final results are distributed among the different nodes. Such a distribution may be desired in various situations including intermediate relations within the execution of a query. In many applications however, the final results are needed in one node. In such cases, for the ring and bucket algorithms the distributed results must be collected and merged at one node. Note that the recursive reduction algorithm already includes the collection time.

Figure 11 shows the execution times including collection. The ring algorithm performs slowest. The bucket algorithm's performance degrades but remains better than that of the recursive reduction algorithm, especially for high uniqueness factors. For uniqueness values below a threshold, recursive reduction is better than bucket. The value of this uniqueness threshold depends on the number of nodes and the data size.

We have assumed that the data with particular uniqueness are generated randomly and that the distribution of different



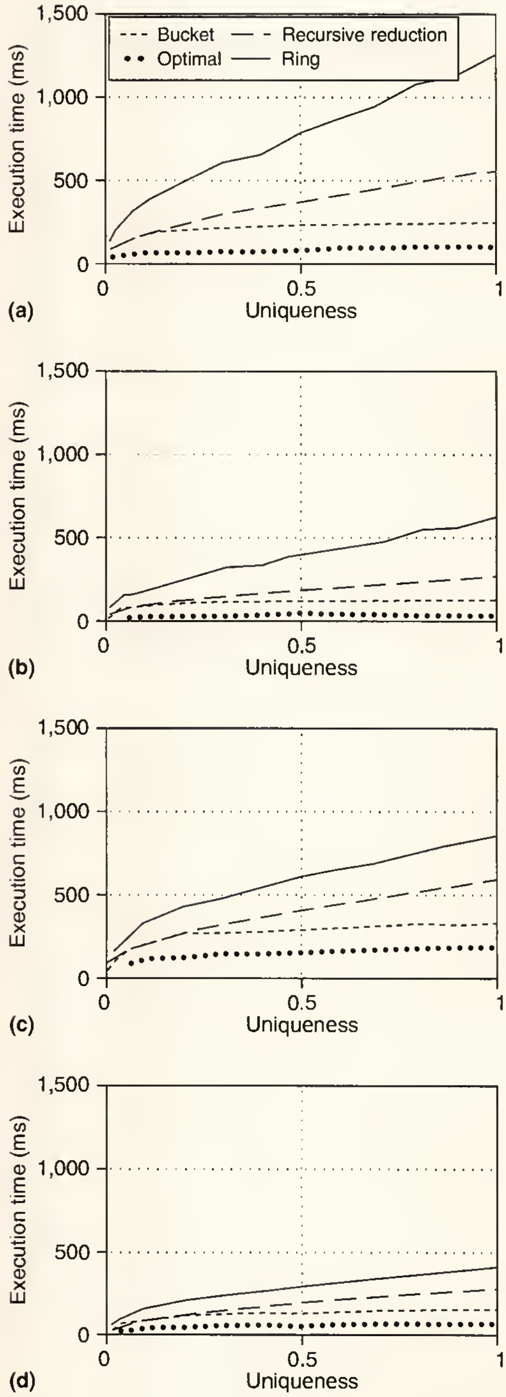


Figure 10. Execution times for low uniqueness, including the local duplicate removal times: 16 nodes, 16,000 data values (a); 16 nodes, 8,000 data values (b); 8 nodes, 16,000 data values (c); and 8 nodes, 8,000 data values (d).

Figure 11. Execution times including collection: 16 nodes, 16,000 data values (a); 16 nodes, 8,000 data values (b); 8 nodes, 16,000 data values (c); and 8 nodes, 8,000 data values (d).

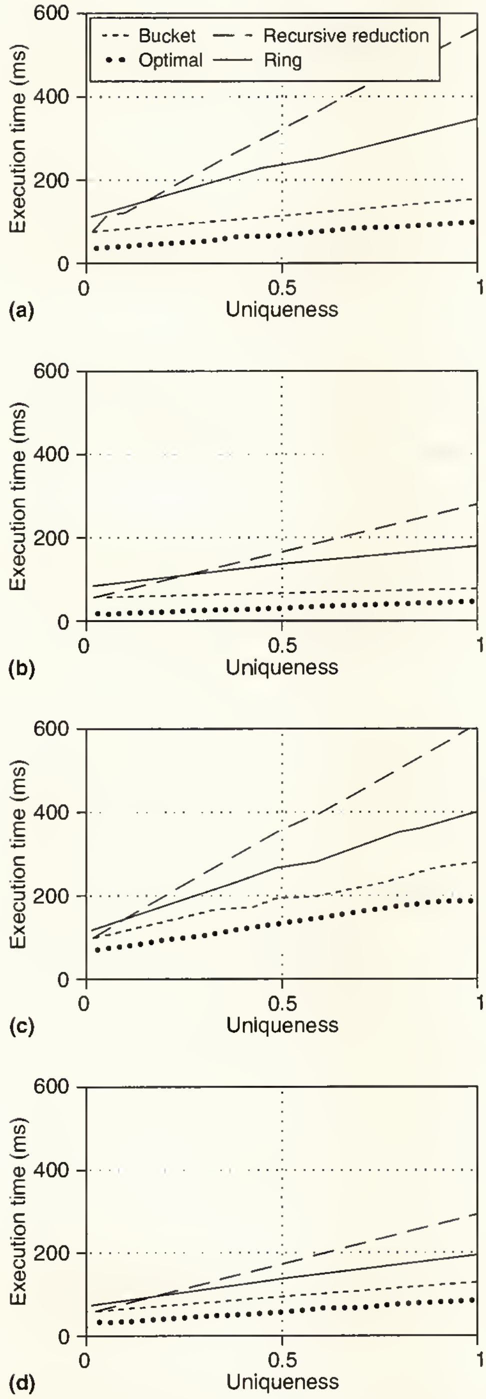


Figure 12. Execution times for nonuniform tuple counts: 16 nodes, 16,000 data values (a); 16 nodes, 8,000 data values (b); 8 nodes, 16,000 data values (c); and 8 nodes, 8,000 data values (d).

unique values in the data is uniform. This assumption may not always be true. We studied the effect of a skewed distribution of the unique data values on the algorithms by generating the data values using a nonuniform distribution. Figure 12 shows the execution times of the three algorithms when we generated data using a Gaussian distribution with μ equaling the mean of the unique values to be generated and σ equaling 50.

As expected, the execution times of all three algorithms remain unchanged for high uniqueness. For low uniqueness, the execution times of the ring and the recursive reduction algorithms decrease. This is not surprising because, if the distribution of the unique values is skewed, the local uniqueness factor on each of the nodes is generally smaller than the ones if the unique values were not skewed. Smaller local uniqueness implies that the local as well as the global duplicate removal times are smaller.

The performance of the ring and recursive reduction algorithms for the uniformly generated data are the worst case performance for a specified uniqueness as far as the skewness of the unique values is concerned. In the case of the bucket algorithm, the local duplicate removal time decreases, as it does for the ring and recursive reduction algorithms.

The global duplicate removal time increases because the design of the bucket algorithm assumes a uniform distribution of data. If the distribution of the data is skewed, the size of each bucket will differ. As a result, after a few iterations some nodes will be processing and transferring large amounts of data, whereas others may be operating on small amounts of data. The discrepancies among the volume of the data on different nodes increase with each iteration. Hence, in the bucket algorithm, the load-sharing among the nodes is not uniform if the data are not generated uniformly. This increases the global duplicate removal time of the bucket algorithm.

The total execution time will either decrease or increase depending on the sum total of the changes in the local and global duplicate removal times. Comparing Figures 9 and 12, we see that the total execution time remains almost constant for the bucket algorithm.

We assumed the distribution of the initial data among the nodes was uniform. This, too, may not be always the case. To study the effect of nonuniformly distributed data on the three algorithms, we uniformly generated data but distributed it nonuniformly to the nodes as follows:

- Generate a random number between 0 and N using a uniform random number generator.
- Let the number be x_0.
- Randomly select x_0 data values and assign them to node 0.
- Now generate a random number between 0 and $(N-x_0)$ using a uniform random generator.
- Let that number be x_1.
- Randomly select x_1 data values and assign them to node 1.

- Repeat this procedure to assign the data to the remaining nodes.
- Assign to the last node all data not assigned to any other node.

As can be expected, the execution times of the three algorithms change substantially every time new data are generated using this procedure. Therefore, we repeated the procedure 25 times instead of the normal five times. Figure 13 shows the results on the nonuniformly distributed data. Note that the execution of all three algorithms increases by a factor of 2 to 3. However, the bucket algorithm remains the algorithm of choice.

Another important issue related to the initial data is the tuple size. So far we have assumed that the data consist of only the key, which is assumed to be an integer value. In reality, a tuple consists of a key along with other fields of varying sizes. In such cases, the duplicates are removed by appending the other fields of a duplicated tuple. For example, assume that a tuple consists of a key and a 10-byte field. If the data contain 50 duplicates of a key value, after the duplicates are removed, the key value will occur only once along with 50 fields of 10 bytes each.

Figure 14 on page 54 shows the execution times of the three algorithms for varying tuple sizes. Note that the execution times of all three algorithms increase by an order of magnitude. The recursive reduction is the most sensitive to the tuple size and clearly performs slowest for large tuple sizes. The sensitivity of the recursive reduction algorithm to the tuple size is consistent with its sensitivity to the uniqueness of p and is explained by examining the three components of the execution times, as discussed earlier. Table 2 on page 55 summarizes our results.

THE EFFICIENT EXPLOITATION OF PARALLELISM in a distributed memory architecture necessitates the development of algorithms designed for the underlying execution environment. Proper algorithmic design accounts for both the computational and communicational demands of the application.

Our study focused on algorithms designed for duplicate removal on a hypercube database system. We proposed, implemented, and evaluated three algorithms. We then used the results obtained by a performance evaluation of these algorithms under varying experimental parameters to determine the optimality of a given algorithm for a particular task.

Modeling the three algorithms and predicting their performance for a given system and load characteristics simplifies the determination of the algorithm of choice.[7] Another advantage of using the analytical models is that the results can be generalized to different-size iPSC/2s, as well as to other types of hypercubes provided certain system parameters—namely, the processing and the communication speeds—are known a priori.

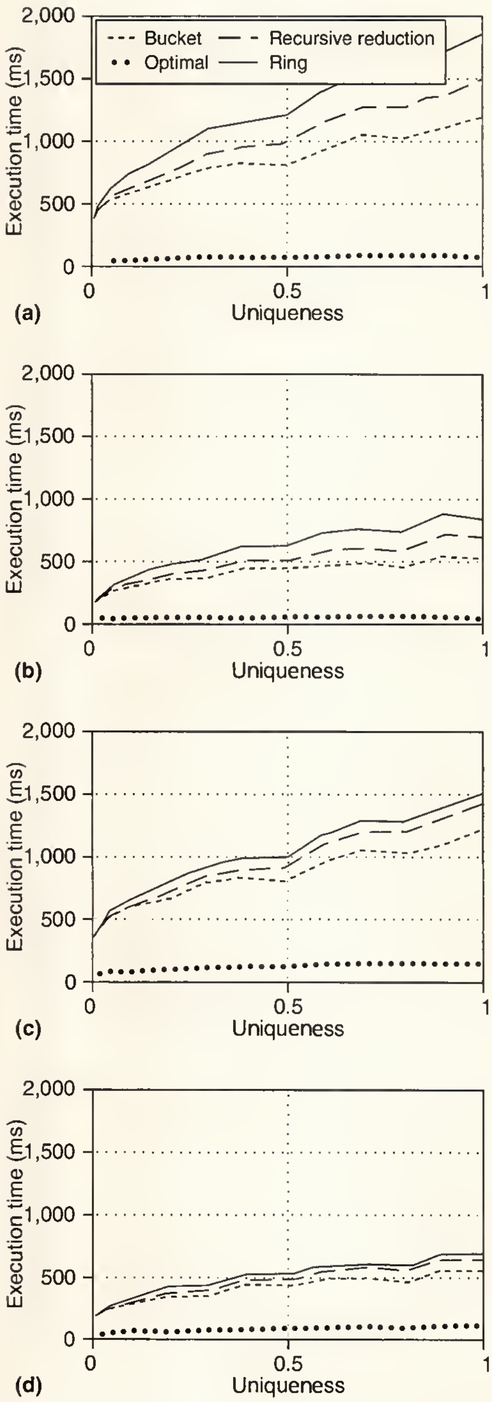


Figure 13. Execution times for data distributed nonuniformly among the nodes: 16 nodes, 16,000 data values (a); 16 nodes, 8,000 data values (b); 8 nodes, 16,000 data values (c); and 8 nodes, 8,000 data values (d).

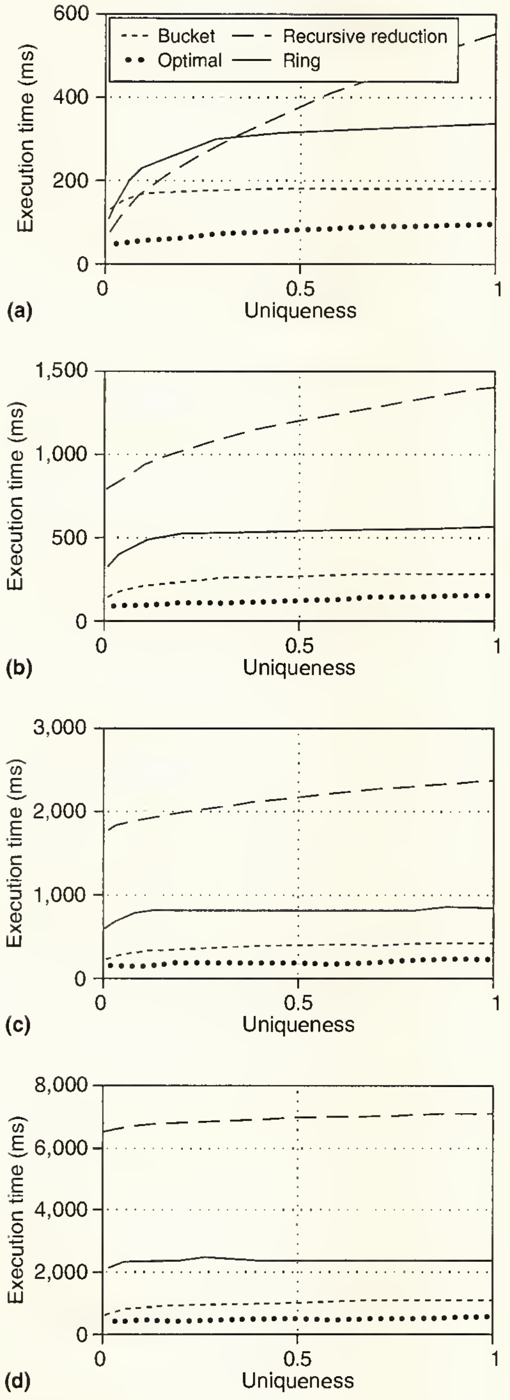


Figure 14. Execution times for different tuple sizes (16 nodes, 16,000 data values): 4-byte tuples (a); 14-byte tuples (b); 24-byte tuples (c); 104-byte tuples (d).

The distributed database is a special case of a multicomputer in which the communication bandwidth is lower than a tightly coupled multicomputer. Therefore, the analytical models of the three duplicate removal algorithms can be generalized to distributed database design as well.

We plan to design a dynamic query optimization scheme that incorporates the results of our study in determining how to best process a given relational operator within a given query. By examining the degree of uniqueness within the input relations and selecting the corresponding optimal duplicate removal algorithm, we intend to reduce the total query processing time. The cost of determining the optimal algorithm is an important consideration that needs to be addressed.

Acknowledgments

Grants from the National Science Foundation (a CISE Research Instrumentation Grant and a Research Initiation Grant), the Virginia Center for Innovative Technology, and Database & Communication Systems, Inc. supported this work.

References

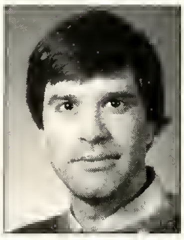
1. C.K. Baru and O. Frieder, "Database Operations in a Cube-Connected Multicomputer System," *IEEE Trans. Computers*, Vol. 38, No. 6, June 1989, pp. 920-927.
2. D. Schneider and D. DeWitt, "A Performance Evaluation of Four Parallel Algorithms in a Shared-Nothing Multiprocessor Environment," *Proc. ACM/SIGMOD Conf.*, Assn. for Computing Machinery, N.Y., 1989, pp. 110-121.
3. W.C. Wong, T. Suda, and L. Bic, "Performance Analysis of a Message-Oriented Knowledge Base," *IEEE Trans. Computers*, Vol. 39, No. 7, July 1991, pp. 951-957.
4. B. Arlid, W. Baugsto, and J.F. Greipsland, "Parallel Sorting Methods for Large Data Volumes on a Hypercube Database Computer," *Proc. Sixth Int'l Workshop on Database Machines*, Springer-Verlag, New York, 1989.
5. G. Fox et al., *Solving Problems on Concurrent Processors*, Prentice-Hall, Englewood Cliffs, N.J., 1988, pp. 327-348.
6. O. Frieder, "Multiprocessor Algorithms for Relational-Database Operators on Hypercube Systems," *Computer*, Vol. 23, No. 11, Nov. 1990, pp. 13-28.
7. V.A. Topkar, O. Frieder, and A.K. Sood, "Duplicate Removal on Hypercube Engines: An Experimental Analysis," *Parallel Computing*, Vol. 17, 1991, North-Holland, New York, pp. 845-871.
8. W.C. Athas and C.L. Seitz, "Multicomputers: Message-Passing Concurrent Computers," *Computer*, Vol. 21, No. 8, Aug. 1988, pp. 9-24.
9. J.P. Hayes et al., "A Microprocessor-Based Hypercube Supercomputer," *IEEE Micro*, Vol. 6, No. 5, Oct. 1986, pp. 6-17.

Table 2. Summary of the algorithm of choice.

Constraints		Algorithm of choice N=16 M=16,000	N=16 M=8,000	N=8 M=16,000	N=8 M=8,000
Initial data not unique and final results may be distributed	Range of data Values known	Bucket for $p > 0.09$ RR* for $p < 0.09$	Bucket for $p > 0.04$ RR for $p < 0.04$	Bucket for $p > 0.02$ RR for $p < 0.02$	Bucket for $p > 0.02$ RR for $p < 0.02$
	Range of data Values not known	Ring for $p > 0.34$ RR for $p < 0.34$	Ring for $p > 0.32$ RR for $p < 0.32$	Ring for $p > 0.25$ RR for $p < 0.25$	Ring for $p > 0.3$ RR for $p < 0.3$
Initial data unique and sorted, and final results may be distributed	Range of data Values known	Bucket for $p > 0.1$ RR for $p < 0.1$	Bucket for $p > 0.04$ RR for $p < 0.04$	Bucket for $p > 0.02$ RR for $p < 0.02$	Bucket for $p > 0.02$ RR for $p < 0.02$
	Range of data Values not known	Ring for $p > 0.4$ RR for $p < 0.4$	Ring for $p > 0.3$ RR for $p < 0.3$	Ring for $p > 0.25$ RR for $p < 0.25$	Ring for $p > 0.3$ RR for $p < 0.3$
Initial data not unique and final results should not be distributed	Range of data Values known	Bucket for $p > 0.1$ RR for $p < 0.1$	Bucket for $p > 0.12$ RR for $p < 0.12$	Bucket for $p > 0.13$ RR for $p < 0.13$	Bucket for $p > 0.13$ RR for $p < 0.13$
	Range of data Values not known	RR	RR	RR	RR
Initial data not unique and final results may be distributed. Tuple counts not uniform	Range of data Values known	Bucket	Bucket	Bucket	Bucket
	Range of data Values not known	Ring for $p > 0.15$ RR for $p < 0.15$	Ring for $p > 0.25$ RR for $p < 0.25$	Ring for $p > 0.08$ RR for $p < 0.08$	Ring for $p > 0.15$ RR for $p < 0.15$
Initial data not unique and final results may be distributed. Data not distributed uniformly on the nodes	Range of data Values known	Bucket	Bucket	Bucket	Bucket
	Range of data Values not known	RR	RR	RR	RR
Initial data not unique and final results may be distributed. Tuple size=29 bytes	Range of data Values known	Bucket	Bucket	Bucket	Bucket
	Range of data Values not known	Ring	Ring	Ring	Ring

RR* Recursive reduction
p Global uniqueness

10. D. Bergmark et al., "On the Performance of the FPS T-series Hypercube," *Proc. Second Conf. Hypercube Multiprocessors*, in *Hypercube Multiprocessors*, SIAM, Philadelphia, 1987.
11. P. Close, "The iPSC/2 Node Architecture," *Proc. Third Conf. Hypercube Concurrent Computers and Applications*, Jan. 1988, SIAM.
12. *iPSC/2 and iPSC/860 Users' Guide*, Report 311532-006, Intel Corp., Beaverton, Ore., June 1990.
13. S.F. Nugent, "The iPSC/2 Direct-Connect Communication Technology," *Proc. Third Conf. Hypercube Concurrent Computers and Applications*, SIAM, Jan. 1988.
14. D. Maier, *The Theory of Relational Databases*, Computer Science Press, Rockville, Md., 1983.
15. L. Allen and O. Frieder, "Exploiting Database Technology in Medical Arenas: A Critical Assessment of PACS," to be published in *IEEE Engineering in Medicine and Biology* (Special Issue on Computers in Medicine), Mar. 1992.
16. O. Frieder, "A Parallel Database-Driven Protocol Verification System Prototype," to be published in *Software Practice & Experience*, Wiley Press, New York, 1992.
17. M. Abduelguerfi and A. K. Sood, "Computational Complexity of Sorting and Joining Relations with Duplicates," to appear in *IEEE Transactions on Data and Knowledge Engineering*. Also available in *Proc. Parbase 1990*, IEEE Computer Society Press, Los Alamitos, Calif., 1990.
18. M.S. Lakshmi and P.S. Yu, "Limiting Factors of Join Performance on Parallel Processors," *Proc. IEEE Fifth Int'l Conf. Data Engineering*, CS Press, Feb. 1989.
19. E. Omiecinski and E. Tien, "A Hash-Based Join Algorithm for a Cube-Connected Parallel Computer," *Information Processing Letters*, Vol. 30, No. 5, 1989, pp. 269-275.

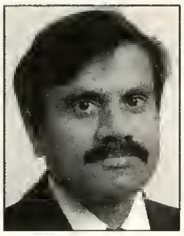


Ophir Frieder teaches computer science at George Mason University. His research interests include parallel and distributed architectures, database systems, operating systems, and medical imaging architectures.

Frieder earned a BS in computer and communications science, and MS and PhD degrees in computer science and engineering, all from the University of Michigan. He is a member of Phi Beta Kappa and IEEE Computer Society.

Vijaykumar A. Topkar is a PhD student in information technology at George Mason University. His research interests include signal processing, computer vision, parallel processing, simulation, modeling, and optimization.

He received a bachelor's degree from the Indian Institute of Technology in Bombay and a master's degree from the Indian Institute of Technology in Kanpur, both in electrical engineering.



Ramesh K. Karne is a PhD candidate at George Mason University. He works as an advisory engineer and scientist at IBM. His research interests are parallel simulation, massively parallel architectures, artificial intelligence and simulation, and intelligent hardware design.

Karne holds a BE in electronics and communication engineering from Osmania University in Hyderabad, India, and an MS in electrical and computer engineering from the University of Wisconisin at Madison. He belongs to the IEEE and the ACM.

Arun K. Sood is a professor of computer science at George Mason University. His research interests include image analysis, signal processing, parallel and distributed processing, database machines, performance modeling, and optimization.

Sood received a bachelor's degree from the Indian Institute of Technology in Delhi and MS and PhD degrees from Carnegie Mellon University, all in electrical engineering. He is a member of IEEE Systems, Man, and Cybernetics Society's Administrative Committee.

Address questions concerning this article to Ophir Frieder, Computer Science Department, George Mason University, 4400 University Drive, Fairfax, VA 22030; or via e-mail at ophir@gmuvax2.gmu.edu.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 159 Medium 160 High 161

Conformance Testing of VMEbus and Multibus II Products

Designers working with the European Community's Conformance Testing Services program have developed a system for testing VMEbus and Multibus II products. Conformance is necessary to allow and guarantee interoperability between products from different vendors. The EC's test system is mainly automated to reduce costs and ensure impartiality. An overview of the system's components familiarizes readers with its procedures.

Marcus Adams

Yi Qian

Jacek Tomaszunas

Computer Research Center (FZI), Karlsruhe

Josef Burtscheidt

Edgar Kaiser

Central Laboratory for Electronics, Julich

Csaba Juhasz

Technical University of Budapest

Conformance testing seeks to determine if a product or parts of a product (in our case, parallel bus interfaces) has been built according to the rules of the applied standard.[1] One way to check conformity is to examine the schematics, programmable logic array data, and PROM data, and compare the results with the rules of the standard. Another way is to create a testing environment representing the standard and test the product against it. Dedicated devices of the test environment monitor the results of interaction.

This second way requires a big investment in methodology, hardware, and software before the first product can be tested. But this approach's advantage is that it produces test results that are objective, reproducible, and independent from the test engineer. A consortium of advisors chose the second method for the Bus Interface Conformance Test (BICT) project.

Test system

The BICT system tests the conformity to standards of bus interfaces of VMEbus and Multibus II boards and their backplanes.[2,3] The test system represents the standards and offers all the options the standards give. This approach allows a board to be plugged into the test system where predefined automatic or semiautomatic procedures test the features of the unit under test.[4]

The bus standards specify four main categories:

- timing and logical behavior of signals,
- message-passing protocol (for Multibus II only),
- electrical properties of boards and backplanes, and
- mechanical dimensions of boards and backplanes.

A specific configuration of the test system and its devices is required for each category. Figure 1 on page 58 shows the general configuration of the BICT system, including the dial bench gauge, which is operated manually.[5,6]

A test system controller manages the activities of the test system devices and gives instructions to the operator, or test officer. For each test campaign the controller collects the measuring results, stores them in a database, and generates a test report on the basis of collected data. The controller performs a series of functions, (see box on page 58) classified by type of action.[7]

In this article we explain a test campaign in the sense of a walk through the test system, in the way a customer–such as a manufacturer, reseller, or original equipment manufacturer–would see it. We focus on testing bus interfaces because it is more complex than testing backplanes.

0272-1732/92/0200-0057$03.00 © 1992 IEEE

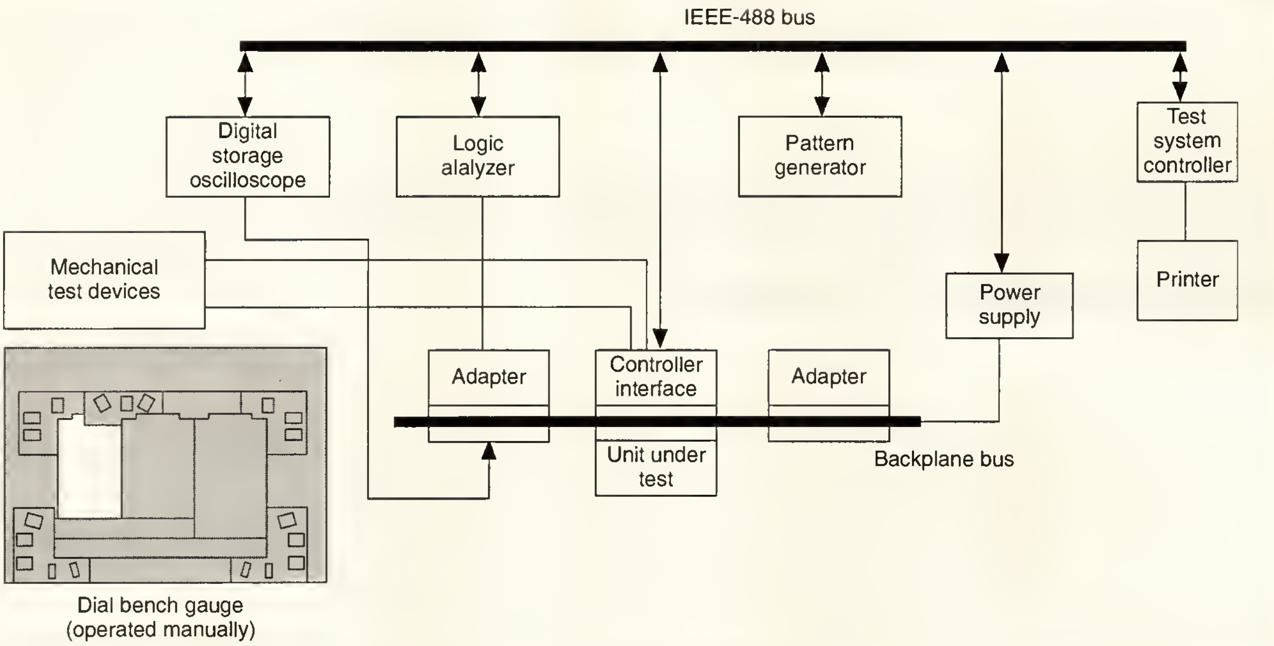


Figure 1. BICT system.

Test system controller functions

For the classifications listed here, the controller provides the following information or performs the following tasks. In the case of manual tests, the operator provides the information, which the controller then analyzes.

The test officer

- Information about the actual test and test state
- Instructions about mechanical tests to be carried through by the officer
- Information about test results by immediate check against parts of the relevant standard
- Requests test results and offers data entry capabilities

Test system controller

- Start/stop logic analyzer
- Start/stop pattern generator
- Poll logic analyzer and digital storage oscilloscope on trigger or time-out if no triggering
- Select and start test routine of the unit's upper tester (via communication channel)
- Upload data from test devices
- Download data to test devices

Digital storage oscilloscope

- Measurement of signal edges and waveforms for electrical test
- Measurement of voltage levels for static electrical test

Logic analyzer

- Run the trigger to monitor the logical and timing behavior of the unit
- Store captured data in case of triggering
- Access to database of trigger setups
- Trigger on certain messages or bus states
- Store captured message in case of triggering

Pattern generator

- Run sequence of patterns for bus operation initiation/response
- Control of delay lines for signal adjustment

Power supply

- Provide unit with current in the specified voltage range
- Measurement of current consumed by unit

Mechanical test devices (dial bench gauge, slide calipers)

- Measurement of boards
- Measurement of devices mounted on boards

BICS/BIXIT

Before testing begins, a customer provides the following information about the unit on two DOS-based, electronic forms:

- **Bus interface conformance statement (BICS).** The customer enters the capabilities and options of the standard implemented in the unit.
- **Bus interface extra information for testing (BIXIT).** The customer provides further details on the implementation features of the unit, such as address ranges, interrupt levels, bus request levels, and message types.

BICS/BIXIT is a menu-driven, interactive input program that automatically checks for completeness and consistency as the user fills in the form. It features dynamic data management and automatic data interfacing and recording. Figure 2 depicts a BICS/BIXIT input screen requesting information about the implemented features of a memory slave.

Chip-level testing is complex, time-consuming, and not cost-effective for the purpose of conformance testing. Board manufacturers use relatively few standardized silicon devices to implement the interface drivers, and sufficient and reliable information about their capabilities is available. We can therefore avoid testing on the chip level.[8]

In lieu of such testing, the board manufacturer provides information about drivers, transceivers, and other devices that implement the bus interface, on another DOS-based form, the vendor claim. The vendor returns this form, along with the BICS and BIXIT data, to the center, where it is analyzed by a test program running on the controller.

In the first step of testing, static conformance review, the controller checks this information against a silicon database and the relevant rules of the standard.

Mechanical test

Test officers manually assist in the next step, testing whether the board fits into a standard rack and its backplane. A mechanical test program running on the controller guides the tester and indicates the points to be measured. The mechanical test is based on two test methods: the outline test with the dial bench gauge and the rack fit test with the rack simulation.

For the outline test, the dial bench gauge (see Figure 1) can be configured according to the controller's instructions and the test officer mounts the unit board onto the gauge. The tester then applies the test tool to the measuring points asked for by the test program. The read values transmit via the IEEE 488 bus to the controller. The program determines if the read value is plausible and compares it with the requirements of the standard.

The dial bench gauge consists of a base plate on which smaller plates are mounted, carrying a reference bolt for each position to be measured. The positions of these reference bolts vary for different board sizes (single, double, and triple European cards, 160 or 220 mm long).

The tester fixes the unit on two clamping strips. Before taking the measurements, the tester samples a reference board to get the offset values. This procedure, called the calibration phase, delivers the offset factors, which are stored in the controller and integrated into the measurement calculation. Figure 3 on page 60 shows the configuration for the mechanical test.

Testers use a slide caliper for special cases where the depth of the gauge is not applicable. The rack fit test is a purely qualitative test, as postulated in the VME standards. The tester performs this test manually.

Electrical test

The electrical test verifies rules concerning requirements for electrical characteristics and behavior. The measurements are electrical, chronological, or a mixture of the two.

One part of the electrical test measures electrical characteristics independent of special capabilities of the unit, such as connectivity, termination networks, and power consumption. The other part times signals or performs specific functions of the unit, measuring signal waveforms (rise and fall times, high and low times), signal levels, and power on/off behavior.

```
                         BICS/BIXIT Edit
File        Edit        View        Check        Document        Quit

                            Master #1
Local Identifier : 68020 CPU
                   Cycle Type

                   Basic Data Transfer . . . . : Y
                   Block-Transfer . . . . . . . : N
                   Read-Modify-Write . . . . . : Y
                   Unaligned Transfer . . . . . : Y
                   Address-Only . . . . . . . . : N

      Local Bus Timer (BIXIT) . . . . : not implemented

Data Width                          Address Width

D8 . . . . . . . . . : Y            A16 . . . . . . . . . : Y
D16 . . . . . . . . . : Y           A24 . . . . . . . . . : Y
D32 . . . . . . . . . : Y           A32 . . . . . . . . . : Y

BIXIT :                                          Remarks :
```

Figure 2. Input screen of BICS/BIXIT.

As with the mechanical test, the controller guides the operator through the electrical tests (see Figure 4). The test officers handle the test probes manually, since for each rule different test points must be checked. The obtained results are uploaded to the controller via an IEEE 488 interface, where a program compares them with the specification values and generates a statement for the report.

The electrical test does not measure driving and receiving characteristics of the signal lines. Testing them would mean testing silicon and, as mentioned earlier, this exceeds the scope of conformance testing on the board level. Therefore, the test officers refer to the information provided on the vendor claim form to check these devices or assemblies. Operators apply this method when items are not testable or may change from board to board. Items tested in this way include used drivers, receivers, and connectors.

Timing protocol test

The timing protocol test determines whether the interface modules of the unit communicate through the bus with other modules in accordance with the relevant timing protocol given by the test specification. (Interface modules include the interrupt handler, interrupt requester, and arbiter.) The system individually tests every module or joined modules of the unit. A pattern generator stimulates the module(s) under test through the bus, while a logic analyzer detects anomal (or, illegal) events on the bus (see Figure 5). The stimulating patterns created on the bus by the pattern generator, as well as the responses created on the bus by the module(s) under test, constitute a bus operation.

The standard rules define the modules and their options, or capabilities. Modules can participate in different bus cycles according to their defined capabilities. For example, a VME master module with 16-bit data transfer capabilities may participate in single- and double-byte read-and-write data transfers, as well as block transfers. But it may not participate in Byte (0-3) data transfers.

A module option may also define a cycle option. A VME arbiter module may resolve a request according to a standard-specific algorithm (single, priority, or round robin) and a VME requester may release the bus according to one of two algorithms (release-when-done or release-on-request).

Bus operation. Every bus operation is specified as a set of bus cycles and hence as a set of timing and data protocol test rules. Part of the bus operation stimulates the test module. The other part is the response of the unit, which will be analyzed by the logic analyzer.

For example, a CPU module reads data and writes it to a

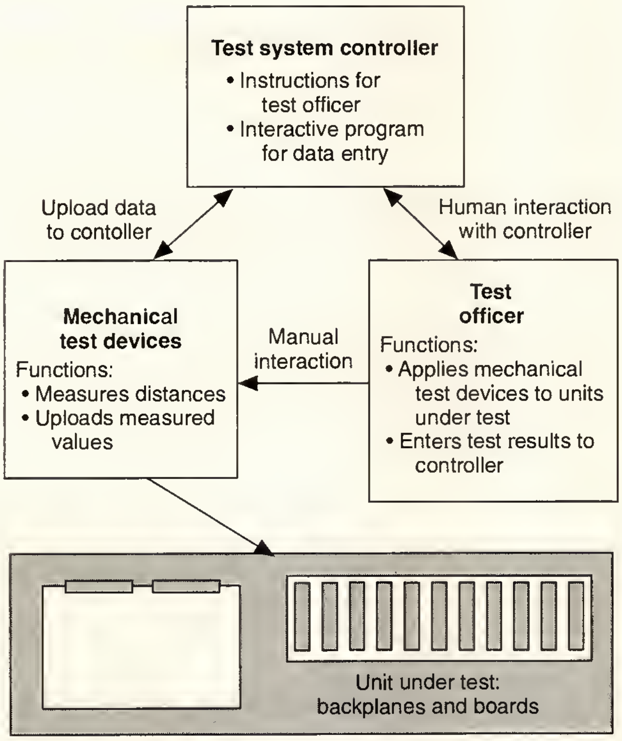


Figure 3. Mechanical test.

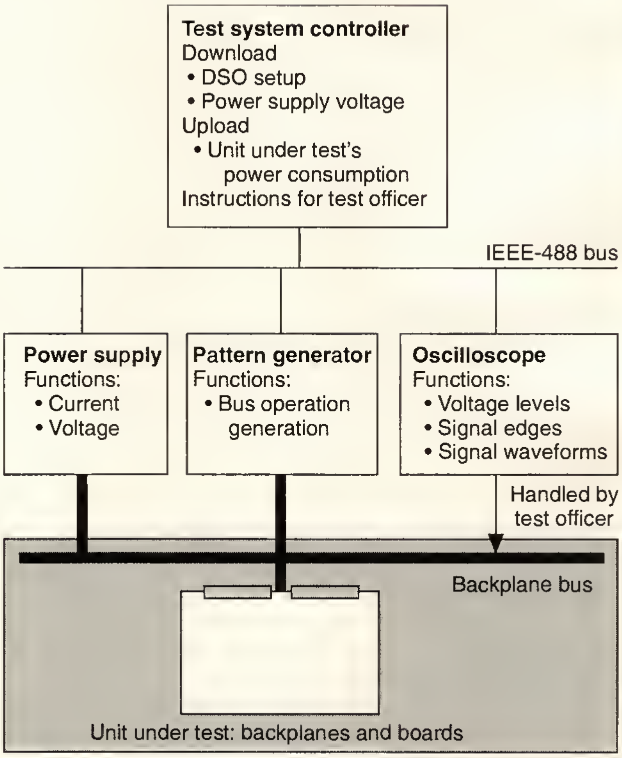


Figure 4. Electrical test.

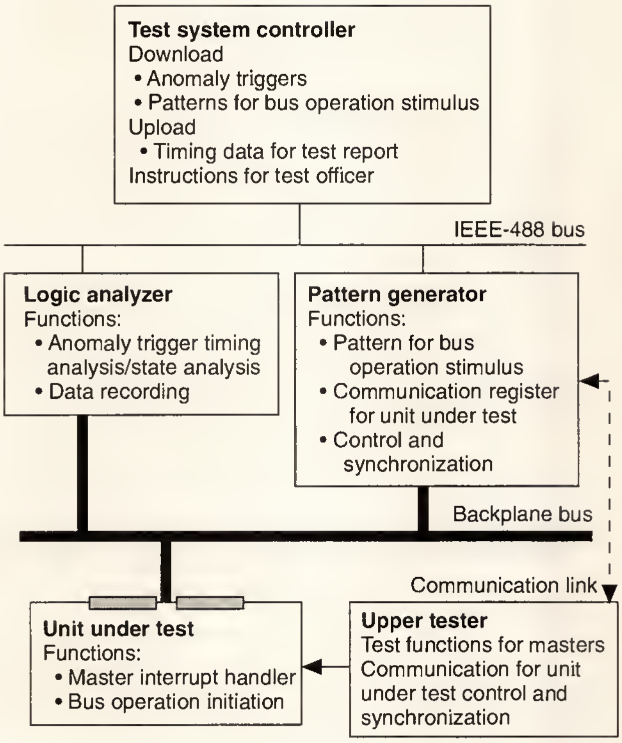


Figure 5. Timing protocol test: test of masters.

memory board. First it runs an arbitration cycle to gain the bus. Next it runs a read cycle to fetch the data from the module. To write its data to the module, it performs a write cycle. Finally, the CPU module releases the bus. In this example the bus operation consists of the arbitration, read, and write-and-release cycles. In every cycle the CPU module (stimulus) gives signals that are driven by the memory module (response).

Pattern generation. The signal relations, into which timing rules are translated, define the limits of the stimulating rules. For worst-case timing protocol testing, we developed worst-case stimulating patterns. For basic interconnection timing protocol testing, the system stimulates the unit using more tolerant limits.

A pattern generation program, reads signal relations associated with a bus cycle as input information and translates them to stimulating patterns and wait statements for the pattern generator. Data protocol test rules are taken into account by the pattern generation program, as needed.

Real-time analysis. The real-time analysis method tests behavioral characteristics (or, functional tests) and verifies the relevant rules. Real-time analysis facilitates exhaustive testing and supports the BICT philosophy:

Many complete bus cycles/operations perform consecutively. The logic analyzer monitors the signals on the bus in real time and only freezes data in its memory if a trigger condition is fulfilled. While cycles run, the controller downloads different trigger words to the logic analyzer.

We decided to run the bus cycles/operations several times because 1) asynchronous behavior leads to deviations in reaction times, and 2) different states of the unit may cause different reactions. This method uses anomaly triggers, so that only in cases where the standard is violated does the logic analyzer upload the data to the controller for the test report.

Control-and-status register test

The control-and-status register set holds system information in a standardized way. During a system startup, the configuration program uses it for systemwide initialization, configuration, and diagnostics.

The test for Multibus II checks the content, structure, and predefined functions of the control-and-status register. Since almost every item tested is a different case, we established an individual test procedure for each rule. During the test these procedures are executed one by one, according to the unit's capabilities.

Message-passing protocol test

The message-passing protocol describes a communication and data exchange procedure between agents using well-defined messages. We call it "network in a crate" because it applies mechanisms similar to those used in LANs. One message packet is transferred using one sequential data transfer bus operation in the message address space. In general, we define two types of messages.

- *Unsolicited.* An unsolicited message is a one-way transfer of a data packet to a destination address. The packet carries up to 28 bytes of data. Systems use unsolicited messages for interprocessor communication, very short data transfers, and virtual interrupts. Within one Multibus II system, we can define up to 255 message (interrupt) sources and 255 message (interrupt) destinations. The broadcast message, a special unsolicited message, is a simultaneous, one-way transfer of an unsolicited message to all available destination addresses in a system.
- *Solicited.* A solicited message transfer is a specified, finite exchange of data packets that transfers up to 16 Mbytes of data. Before such a message is sent, the system negotiates to ensure there is enough free data buffer available at the destination and to define other transfer parameters. The solicited message fragments into small

data packets, each containing at least 32 bytes. The system then sends them over the bus, one at a time.

The message-passing protocol test for Multibus II checks that the unit performs the sending and receiving of unsolicited and solicited messages correctly. For solicited messages, it also tests the negotiation of the transfer and the correct application of the negotiated parameters. In addition, the content of each message is checked.

Since the message-passing protocol uses a bus protocol on signal level, the unit must pass the electrical and functional test (on signal level) successfully before operators perform the message-passing protocol test.

To perform this test, the vendor installs an upper tester which substitutes for the software layer above the unit. Figure 6 shows that a test routine runs through a communication channel between the controller and the upper tester.

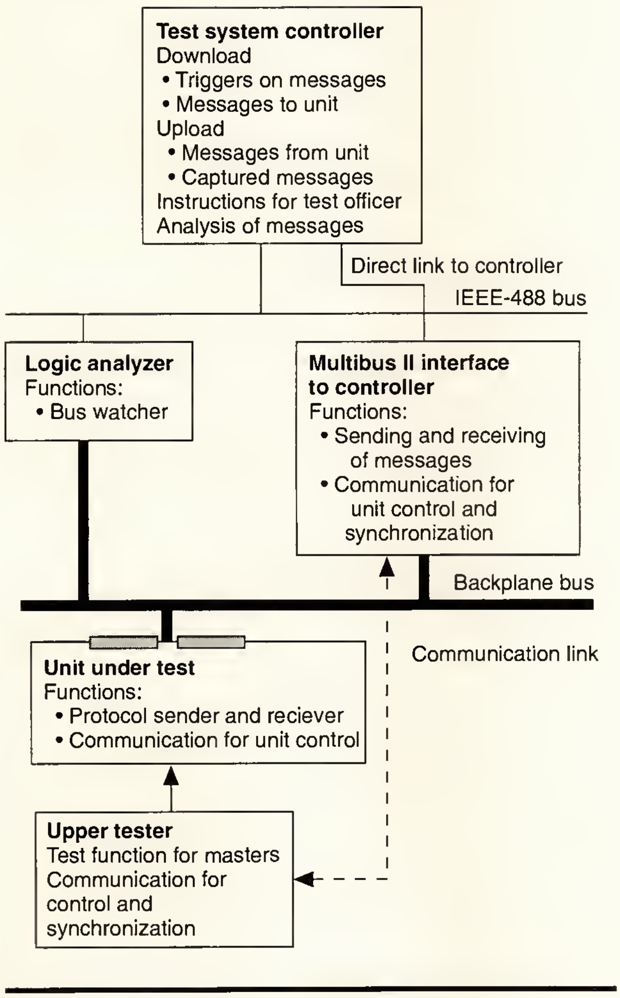


Figure 6. Message-passing protocol test.

BICT divides the message-passing test into two entities, each applying different analysis methods.

- *Messages sent.* The controller uses two methods to analyze outgoing messages. In most cases, the system uses a software analysis method. The Multibus II interface receives the unit's messages, which are then analyzed by the controller. But in some cases, in which the logic analyzer must monitor the bus, the system uses a bus analysis method. This method measures, for example, the duty cycle during a solicited message transfer, or detects unexpected messages sent by the unit.
- *Messages received.* Two methods can be used to test the message-receiving capability of a unit. In the on-board message verification method the upper tester analyzes the messages received by the unit and reports the test result to the controller. Since this will blow up the upper tester and with it the vendor's investment in installing it, we prefer the message reflection method. Here, the messages sent to the unit are reflected by the unit and analyzed by the test system.

To ensure that detected errors are caused by the unit's receiving capability and not by its sending capability, the unit must pass the sending capability before the test officer applies the message reflection method.

Test example

To demonstrate how the conformance test is carried out in practice, we describe here a message-passing protocol test case for Multibus II. Rule 13.5.1.2, Section 5 of the Multibus II test specification states, "If an agent is not able to receive a requested solicited message, it must send a buffer reject message to the agent that has sent the buffer request message." This rule checks the negotiation of a solicited-message transfer. The buffer reject message is an unsolicited message that sets up a solicited-message transfer. A system refuses the buffer request if, for example, a receiver does not have enough resources or the resources are, at that moment, not available.

Verification procedure. To test this rule, we apply the software analysis method. At the beginning of the verification procedure the controller, via the communication channel, starts a test routine of the upper tester installed on the unit. The selected test routine is responsible for receiving solicited messages.

Then the controller, via the Multibus II interface, sends a buffer request message to the unit (see Figure 7). In this case the requested length of the solicited message, 10 Mbytes, is greater than the available local resources of the unit, 1 Mbyte. The size of the unit's local resource has already been declared on the BICS/BIXIT form, so the controller can automatically calculate the requested length of the message.

Now the controller waits until the Multibus II interface re-

ceives a message. If no message arrives within a specific time, the test of the rule has failed. If a message arrives, the controller verifies whether the message is a buffer reject.

Results. At the end of the verification procedure the controller interprets the results and appends them to a log file. In the case of rule 13.5.1.2 section 5, it assigns a "failed" result if the unit sends no message or a message that is not a buffer reject.

Test report. After the test, the center prepares a report (see Figure 8) and sends it to the customer. The report describes the unit, the test suite, and all useful information concerning the executed tests and their results. The report's first section summarizes the number of tests applied and failed. A later section details the applied tests and the test sequence. If a rule was violated, the report notes the error and gives the corresponding rule text. A description of the test method applied can be found in the test specification. Customers can obtain more detailed data on the test execution and results, such as captured waveforms or data, from the test laboratory if needed for further analysis.

THE BICT SYSTEM OFFERS vendors the security of an impartial test of a board's conformity to bus standards. Conformity is a necessary precondition for the interoperability of boards sharing a bus.

To test a product, a center needs information only about the features and parameters of the board, not design information such as schematics, PROM, or PLA data. Design information stays with the originator.

The mainly automated test procedures ensure cost- and time-effectiveness and guarantee objective results. A basic aim of the BICT project was to eliminate as much human influence as possible. The results depend only on the strengths and weaknesses of the test system. We are optimistic that any remaining weaknesses will be eliminated as the test system matures.

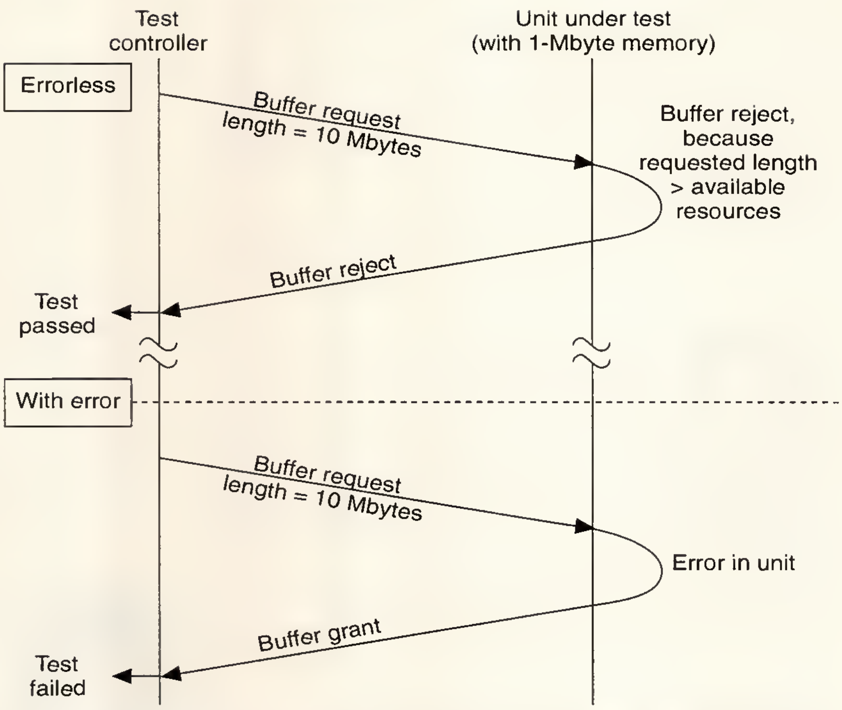


Figure 7. Test of negotiation phase (unit as receiver).

Message passing protocol test
The following errors were identified:

Rule number	Start Date	Start Time	End Date	End Time	Result
13.4-1	910522	083910	910522	083910	passed
13.5.1.2-4	910522	083910	910522	083911	passed
13.5.1.2-5	910522	083956	910522	083957	failed
*** buffer grant message received from UUT					
13.5.1.2-7	910522	084002	910522	084003	passed

Details of failed tests

The following test cases failed:

Rule 13.5.1.2-5

If an agent is not able to receive a requested solicited message, it must send a buffer reject message to the agent that has sent the buffer request message.

Figure 8. BICT report section.

Starting this year, the service will be available at the test centers involved in the BICT project: the VDE Test and Certification Institute in Offenbach, Germany; S. Seferiades and Associates (SSA) in Athens; and Istituto Italiano del Marchio di Qualità (IMQ) in Milan.

Acknowledgments

The European Community's Conformance Testing Services Program, Phase 2, (contract number 87/12227) sponsored this project.

Marcus Adams heads the microcomputer technology department of the Computer Research Center (Forschungszentrum Informatik) in Karlsruhe. In 1986 he founded the German chapter of the VMEbus users association.

Adams received the Diploma in electrical engineering and a PhD in informatics from the University of Karlsruhe. He is president of the German VMEbus chapter and belongs to the German Electro- technicians Commission and the Association of German Electrotechnicians (VDE).

Yi Qian is a visiting scholar at FZI. His research interests include microprocessor systems, bus conformance test, pattern generation, and automatic test and measurement.

Yi holds a BS in electrical engineering and an MS in computer engineering from Peking University for Posts and Telecommunication, where he is a PhD candidate.

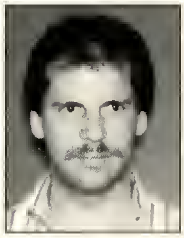
Jacek Tomaszunas is a visiting scholar in the department of microcomputer technology at FZI. He earned the Diploma and MS degrees in electrical engineering from the Technical University of Wroclaw in Poland.

Josef Burtscheidt works on the BICT project at the Central Laboratory for Electronics at KFA Research Center in Julich. He has developed and designed experiment control and microprocessor systems.

Burtscheidt holds a Diplom Ingenieur in communications and information processing from the Technical High School of Cologne.

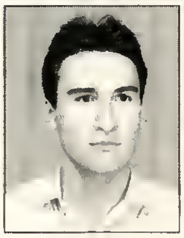
References

1. *Information Processing Systems - OSI Conformance Testing Methodology and Framework*, ISO/IEC DP 9646-1,-5, Part 1, 1988-01-09 and Part 5, 1988-06-22.
2. *IEEE 1296 Standard for a High-Performance Synchronous 32-Bit Bus: Multibus II*, 1988-02-08.
3. *IEEE Standard for a Versatile Backplane Bus: VMEbus*, ANSI/IEEE Std 1014-1987, 1987-09-11.
4. BICT consortium, "Test Methodology," Tech. Report, European Commission, Brussels, Feb. 1990.
5. BICT consortium, "Test Tool Description," Tech. Report, European Commission, Brussels, Feb. 1990.
6. BICT consortium, "Final Test Tool Description," Tech. Report, European Commission, Brussels, Apr. 1991.
7. BICT consortium, "Test Bed Description," Tech. Report, European Commission, Brussels, Apr. 1991.
8. BICT consortium, "Board Test Procedures," Tech. Report, European Commission, Brussels, Oct. 1990.



Edgar Kaiser also works on the BICT project at the Central Laboratory for Electronics. He, too, has developed and designed experiment control and microprocessor systems.

Kaiser earned a Diplom Ingenieur in electronics and information processing from the Technical High School of Aachen.



Csaba Juhasz is a PhD candidate in process control at the Technical University of Budapest. He served as a research assistant on the BICT project at FZI.

Juhasz earned an MS in electrical engineering from TU, Budapest. He is currently a visiting scholar at the Department of Systems Science at The City University in London.

Address questions concerning this article to Marcus Adams, Forschungszentrum Informatik, Computer Research Center, Haid-und-Neu-Strasse 10-14, W-7500 Karlsruhe, Germany; or via e-mail at adams@fzi.uka.de.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 162 Medium 163 High 164

How I spent my Christmas vacation

Richard Mateosian

2919 Forest Avenue

Berkeley, CA

94705-1310

(510) 540-7745

My holiday tradition is to attack the big projects that I've been putting off all year. This year I plunged into the Macintosh.

I have a Mac SE/30. I use it every day in my consulting work. I also use it for reviewing the Macintosh software that I receive in abundance. This is the source of the problem.

In my consulting work I do simple word processing. I occasionally transmit text files over phone lines. My Mac SE/30's original memory size of 2 Mbytes was more than adequate for these tasks.

I enjoy reviewing new software, since the process forces me to learn to use powerful new tools that my natural inertia would otherwise keep me from exploring. Unfortunately, new software grows by leaps and bounds, and I can't keep up with it unless my computer does the same. Thus, my project for the holidays was to upgrade my Macintosh to 8 Mbytes.

Memory upgrade

Looked at objectively, a Macintosh memory upgrade is a pretty small project. It takes less than half an hour, but somehow I managed to put it off for nearly four months after I had everything I needed in hand. I probably would never have done it at all if it had not been for the idiot-proof package put together by the mail-order company called Mac Warehouse.

I called Mac Warehouse's toll-free number (800) 255-6227 to order the 1-Mbyte single in-line memory modules (SIMMs) required to upgrade a Macintosh. As I write this, their advertised price is $59 each. I needed eight of them to upgrade from 2 Mbytes to 8 Mbytes. Each one replaces one of the original eight 256-kilobyte SIMMs. The Mac Warehouse people told me I needed a simple toolkit, including a grounding wrist strap, which they sold me for $9. They included a video cassette demonstration of the upgrade process. The entire package showed up at my house the next day. Then it sat in my office for more than three months.

The video is marvelous. I am about as unmechanical as a graduate of an engineering school could possibly be. Removing the SE/30's motherboard would have been beyond me with only written instructions. A woman with a reassuring manner demonstrated all of the simple but hard-to-explain-in-print steps required to upgrade any Mac's memory. With exquisitely manicured hands she deftly manipulated tools, eased wire harnesses out of hard-to-reach sockets, and snipped a resistor lead. She carried on a running description of what she was doing, calmly mentioning the inherent dangers at certain points. My favorite was when she said, "If you crack the video tube, it *will* implode."

I carried out the entire task in about a half hour. I sat on my living room rug with the grounding strap on my left wrist, the VCR remote control unit by my right hand, and the TV, my Macintosh, and my tools in front of me. The video runs 11 minutes, but I kept replaying key parts until I was sure I understood what I was doing. When I reassembled my Mac, everything worked perfectly.

Operating system upgrade

I would never change my operating system software if I didn't have to. Unfortunately, new software invariably makes old operating systems obsolete. When I loaded the latest version of Mathematica onto my Macintosh, it immediately announced that it wouldn't work on the version of the Macintosh operating system that I was

0272-1732/92/0200-0065$03.00 © 1992 IEEE

running. I could have put in a minor upgrade from 6.03 to 6.07, but I decided to switch to System 7, Apple's long-awaited major upgrade.

The Berkeley Macintosh Users' Group (BMUG) is a wonderful resource for anyone who uses Macintosh computers. When I decided to upgrade to System 7, I called them at (510) 549-BMUG. I bought their package deal, which includes the 10-disk upgrade set from Apple and *The Little System 7 Book* by Kay Yarborough Nelson (Peachpit Press, Berkeley, 1991, 158 pp., $12.95), all for $25. I did this about the same time I bought the SIMMs, and everything sat around my office for just about as long.

System 7 is quite different from previous Macintosh operating systems, but by the time I sat down to use it I felt right at home. I attribute this to the quality of the upgrade package that Apple put together, the excellence of *The Little System 7 Book*, and the many informative articles about System 7 in the *BMUG Newsletter*.

Users have come to expect installation programs to guide them through the installation of large application packages. While installation programs have certainly improved considerably over the last few years, Apple's upgrade software is a cut above any other that I've seen. It is a model for others to follow. Especially impressive were two Hypercard stacks designed to familiarize users with the new features of System 7. The one dealing with the new networking features is especially impressive. It simulates the behavior of the system, allowing you to make menu selections, click buttons, and so on, just as you would if you were performing the networking functions it is teaching you.

While my move to System 7 was easy, it also had its rough spots, all of which hinge on compatibility. Because the underlying software of System 7 differs substantially from previous versions, many software packages designed for previous versions will not run. The Apple upgrade program begins by printing a list of all of the applications on your machine. It labels each one as "fully compatible," "mostly compatible," "must upgrade," or "information not available." Unfortunately, the last two categories predominated on my machine. The only ones listed as fully compatible were Word, Excel, Mathematica, and Mac Draw II.

System 7 differs from previous Mac operating systems; but I felt right at home with it.

The compatibility printout contains helpful information on upgrading old software. It lists your version number and that of the first compatible version. It gives the phone number of the manufacturer of each package you need to upgrade. In some cases it indicates that the upgrade package actually includes a compatible version. Unfortunately, it appears to say that the upgrade package includes a compatible version of Hypercard, but this is only true if you buy the System 7 Personal Upgrade Kit from Apple. My BMUG package did not contain Hypercard, and the Hypercard version that came originally with my SE/30 is incompatible with System 7. So I had to scurry around to obtain a working version.

While simple incompatibility with the programming of System 7 is a problem for most packages, some have an even worse problem. System 7 makes a large part of what they do obsolete. The popular Suitcase II and Master Juggler programs are prime examples. Many of their capabilities for managing fonts, sounds, and desk accessories are no longer needed. System 7 handles sounds and fonts by allowing you to drag their icons into or out of the system file. It eliminates the distinction between applications and desk accessories altogether, since it now incorporates the features of Multifinder. The apple menu simply lists the contents of a folder in the system file. The apple menu folder can contain any application, document, or folder. Selecting an item from the apple menu causes the item to be opened.

Another important feature of System 7 makes front-end programs like Power Station less important. System 7 allows you to create an alias for any document, application, or folder. The alias consumes a tiny amount of disk space and serves as a pointer to the actual item. Thus, for example, you can leave Excel or Word in its own folder but place an alias for each in the apple menu folder or the start-up folder. You can make aliases for the apple menu folder and the system file and leave the aliases on the desktop. I made an alias for the text file that contains this column and placed it in the apple menu folder. When I select "Column" from the apple menu, the system opens this text file and the corresponding word processor.

One important Power Station function that I haven't figured out how to do in System 7 is to associate an icon with an application and a specific set of documents to be opened with it. I would certainly keep Power Station under System 7 for its document-handling capabilities alone, but unfortunately my version caused the system to crash when I tried to run it. The Apple compatibility checker provided no information about Power Station, so this will take additional work on my part. I encountered a number of other small nuisances like that during the upgrade, but on the whole I'm quite happy.

As I write this column using Microsoft Word running under System 7, I am simultaneously also running Mathe-

matica, Hypercard, and the Variable Symbols Mathematica Help Stack (see Micro Review, Aug. 1991). I even have another 250 Kbytes left over in my new 8-Mbyte memory. I'm impressed. I wonder how long this euphoria will last.

Books

The holidays are also a nice time to sit by the fire reading, so I managed to look at a number of good books. One of the best of these (see next paragraph) is a little old for a computer book, and it does show its age in places, but it is so good that it's worth reading anyway. I hope the publishers bring out an updated version.

The Sachertorte Algorithm and Other Antidotes to Computer Anxiety, John Shore (Viking, New York, 1985, 286 pp., $16.95)

Shore states his purpose to be promotion of a general understanding of computers. As a reflective and witty old hacker, he is well qualified to achieve that purpose. It looks to me as though he has done so, although I can't look at the book through the eyes of the intended audience. All I can say is that again and again he makes points that I'd want to make to an interested and intelligent beginner. It's hard to communicate the overall effect of this by quoting isolated phrases, but here are a few that appealed to me:

> In general, jargon-filled error messages are symptoms of a basic problem, namely that the designers and programmers of many office and personal computer systems have failed to separate their own concerns from those of the user.

> If you buy a new car and spend the next year having the dealer fix things that didn't work right to begin with, you don't say that your car is being maintained; you say you bought a lemon. By this criterion, most software products are lemons.

> While standing recently in a cold shower, I had occasion to think about replacing my hot water heater. Because I had been thinking about software engineering before the water turned cold, I was impressed by the extent to which I could choose a new water heater without thinking about air conditioners, telephones, windows, or practically anything else except the number of people in the house and the number of dollars in my bank account.

The title of Shore's book derives from the extended example he gives of trying to use his mother's recipe for Aunt Martl's Sachertorte. He illustrates the process of stepwise refinement as he moves from his mother's original instructions (prepare ingredients; bake cake) to a more detailed sequence of steps that are all within his own more limited repertoire of cooking operations.

Shore wants his readers to understand that programming computers is an exercise in careful and precise communication. The programmer must communicate with the machine, but even more importantly, the programmer must communicate with other human beings. In this sense, programming is a literary activity. When seen this way, some of the problems of the "software crisis" are easier to understand. Writing is easy, but writing well is hard.

Shore also wants us to look at programming as mathematics and as architecture. Here he laments the fact that millions of people are acquiring personal computers and programming them with the languages and techniques of the sixties. The concepts of abstract specification and information hiding that were introduced in the early seventies are only beginning to be widely used today. Proofs of correctness, long recognized to be theoretically possible and highly desirable, are still no more than classroom exercises.

The tools of the writer, the mathematician, and the architect are the keys to managing the complexity of large software packages. The failure to manage complexity, in Shore's view, is the cause of our current software crisis. Shore does a good job of explaining and illustrating the problems of complexity and the failure of our tools in terms that should be accessible to readers of all backgrounds.

If you see this book in your local bookstore, buy a copy to give to a nontechnical friend. You might enjoy reading it yourself first.

Mathematica: A Practical Approach, Nancy Blachman (Prentice Hall, Englewood Cliffs, N.J., 1992, 380 pp., $30)

I wanted to install 8 Mbytes of memory in my Macintosh so I could run Mathematica comfortably. Unfortunately, inadequate memory is not the only obstacle to using Mathematica. Mathematica is a large program with many capabilities, and before Blachman's book we had no adequately tutorial introduction to it. Stephen Wolfram, the creator of Mathematica, wrote a book that is supplied with the program, but as Blachman points out, learning Mathematica from that book is like learning English from a dictionary.

In my August 1991 column, I discussed some of the work Nancy Blachman and her company, Variable Symbols, have done to teach people to use Mathematica. She based the current book on undergraduate courses she taught at Stanford University. It draws on her extensive experience with the difficulties people encounter in learning Mathematica.

I'm impressed by the quality of this book. Blachman provided Prentice Hall

with camera-ready copy, so she can take credit for its attractive appearance and its excellent editing. Mathematica can also take some credit, since the book began as a Mathematica notebook, and many of the illustrations were originally done using Mathematica.

Learning Mathematica is not my top priority, so working my way through this book will be a background project for a while. In my next column I'll tell you how it's going. So far, I like Blachman's approach.

The Parents Guide to Educational Software, Marion Blank and Laura Berlin (Microsoft Press, Redmond, Wash., 1991, 424 pp., $14.95)

The authors of this book are an educational psychologist and a developmental psychologist. Their intended readers are parents who want to give their children the advantages of computer-assisted education but don't know what to do next. As the authors point out,

> Currently you can choose from over 10,000 programs, covering almost every subject imaginable. And, as always, excellence is rare. If you're like most parents, you probably don't see this array of options as a pool of resources but as a mire of confusion.

The authors have selected more than 200 programs for careful rating. They have applied several criteria to this selection, and some of these criteria differ from those used in selecting programs to be used in a school setting. Basically, all of the programs in the book are of high quality, attractive to children of the intended age group, sufficiently varied that children won't tire of them immediately, and inexpensive enough for many families to consider buying them. Where possible the authors tried to select programs that are helpful to the 10-15 percent of the students who have learning difficulties in specific learning areas.

Most of the book is devoted to thorough reviews of the selected programs, but the authors begin with helpful introductory material. They explain how computers can help with learning. Then they review the major outlines of the elementary school curriculum, focusing on what skills are required by each area of study. Finally, they talk about how to set up an educational computer center at home and how to help your child use the programs that you acquire.

The reviews all follow a common format. The authors identify each program as being for a specific age range and school grade range. Then they summarize the hardware requirements, the abilities a child must have to use the program, and the curriculum areas that the program addresses. A detailed discussion of the program itself follows these short summaries.

I have no experience with any of the programs the authors have included, so I can't judge their reviews. However, the reviews that I read addressed issues that I'd want to consider in buying educational software. One interesting example was a $14.95 program for children in the two-to-four age range. The child would not be able to start the program because of a complicated copy protection scheme, so the parent would always have to be there to start it. The authors emphasize this point without passing judgement. I admire their restraint.

I think this book is a really good investment for anyone considering buying educational software.

Annual phone list update

My year end would not be complete without the annual overhaul of my phone list. This year I finally automated the process.

Address Book Plus (Power Up Software Corp, 2929 Campus Drive, San Mateo, CA 94403, (415) 345-5900, $99.95)

This program provides what most people need to handle their little black books. It lets you build a small database, define selection and sorting criteria, and generate reports in a variety of useful formats. These features are mostly built in, so that users don't need to use query or report generation languages.

The program supports printing formats corresponding to the popular pocket and desk appointment books. It also has a special Instabook format, which lets you print a stack of two-sided pages and turn them into a pocket-size address book, simply by cutting, folding, and stapling. I tried it, and it really works.

The program offers an integrated telephone dialing facility. It seems to have all of the flexibility necessary to deal with prefixes, area codes, choice of long-distance carrier, international dialing, modem control, and so on.

Power Up has gone to great lengths to make this program fit into your way of doing things. It can import or export files in formats that you define, and it has built-in formats allowing it to import from Hypercard and to export to personal organizers from Casio and Sharp. It also works with Power Up's mail merge program (Letter Writer Plus).

It's taken me a long time to get around to using a program like this, because no program comes close to doing all the little things I do by hand, but this one is good enough. What it does for me outweighs the things I'll have to give up.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 183 Medium 184 High 185

Micro Standards

Understanding the Acronym Tower of Babel

Carl Warren

McDonnell Douglas

Space Systems Company

(714) 896-3311

x. 7-1230

warren@ssdunx.mdc.com

I am often asked what a particular acronym means, or where it came from. After reflecting on these questions, I realized that they might be the meat for an interesting column. After all, we use these "shortened" words to convey large amounts of information within a standard, or other technical documents.

Interestingly, many of the acronyms and definitions used today came about because of the telegraph and Teletype era. Since telegraphers had to tap in each letter of a word sent, they naturally wanted to use the smallest words possible. Consequently, a unique shorthand was created. This shorthand carried over to Teletype users, partly due to the slowness of the system and the difficulty of typing on the keyboard. Any of you that remember a KSR 35 will understand what I mean.

Interestingly, World War II and the Korean conflict also contributed to the acronym pool. The idea was to send as much information as fast as possible so the enemy wouldn't have a chance to locate the signal or decipher it.

Another interesting note concerns the contributions industry has made to the acronym Tower of Babel. The airline industry, in days before sophisticated reservation systems, used a form of shorthand called Fast Talk. This method let an agent in Chicago call an agent in Des Moines to reserve a seat through to Los Angeles.

Even with the advent of computer systems and better communications than Teletype, the airline industry developed a worldwide standard abbreviation system called SIPP (Standard Information Processing Protocols) codes. This system of codes allowed a reservations agent at United Airlines to send a great deal of easily understood information about a passenger to another carrier. For example, C-19 meant that the passenger required special handling. Other codes indicated special food or an unpleasant passenger.

Similarly, the airlines also had a system for identifying lost baggage. For example, 02 Blue indicated a blue, hard-sided Samsonite bag and 03 denoted an American Tourister bag. Type 20, on the other hand, was a folding garment bag. I was one of the developers of United Airlines Total Apollo Baggage System (TABS) that made use of this information to automatically search for lost bags—a system that ultimately saved United several million dollars a year in mishandled baggage.

Of course, NASA (the US National Aeronautics and Space Administration) probably gets the prize for developing acronyms. It isn't unusual to pick up a NASA-developed document and find that it can't be read without the help of an acronym glossary close at hand. NASA's rationale is that so much information has to be conveyed that abbreviations, acronyms, and initialism (AA&I) enhance the readability of the document.

Because of limited space, I've taken two approaches in my acronym/definition list beginning on the next page. I've chosen some interesting acronyms and expounded upon them, and others I've just listed with their meaning. I could write volumes on this area alone or include multiple meanings, but I won't. I covered lots of ground with this column. You can probably guess, however, that I only touched the surface. If you have a favorite acronym that you would like to share and can tell a short (no more than two-paragraph) story, I'll include it in forthcoming Micro Standards columns.

Finally, some parting thoughts: Didja [sic] know that Soroc, the now-defunct spin-off terminal manufacturer from Lear Siegler, used an anagram for its name? Take a close look at Soroc, and you'll

continued on p. 72

0272-1732/92/0200-0069$03.00 © 1992 IEEE

Glossary

Ack: acknowledgment, an old communications term that was used with Teletype systems. The telegrapher on the receiving end would indicate that a connection was made, or a message received, either by sending AK (later ACK) or their initials.

ADC: analog-to-digital conversion.

AIIM: Association for Information and Image Management.

ANSI: American National Standards Institute, the governing body for the management and creation of standards (see IEEE, MSC, SPARC, and TCCM).

AT: Advanced Technology, an IBM-fostered term describing the basis of the PC bus architecture. This architecture built on the 8-bit and 16-bit PC and XT (Extended Technology) buses of IBM's earlier machines.

Ata: at attachment.

BER: bit error rate, an important metric when referring to storage or communications devices, which gives the number of bits at fault during various types of measures of the device.

BSR: Board of Standards Review.

Bsy: busy. Again, this is a term from telegraphy days. When queried, the downline telegrapher would respond with a busy reply, requesting a hold on traffic. Later with Teletypes, a busy signal was sent out much like the busy signal on your telephone being high, thus denoting busy, or setting of a "busy" bit in a UART, for example.

CAM: common access method/ content access memory.

CBEMA: Computer Business Equipment Manufacturers Association.

CISC: complex instruction-set computer, a class of processor that has a large number of instructions to choose from. An example is the Intel 80386 or Motorola 68040. (See RISC).

CKD: count-key-data.

CR: connection resource.

CRC: cyclic redundancy check.

CS: continuous servo.

DADI: Directly Addressable Device Interface.

DASD: direct access storage device, a disk or other secondary storage device that permits access to a specific sector or block of data without first requiring the reading of the blocks that precede it.

DMA: direct memory access, transfer of data between the memory and a peripheral (or another memory) without intervention of the CPU (besides DMA controller initialization).

DOTS: digital optical tape system.

DTA: dedicated test article. Various test units may be used for various purposes in a test enviornment. A DTA is specifically dedicated to test purposes and as such may have special test ports and special power inputs, and the cabinets may or may not exist.

DUT: device under test. This refers to the actual hardware that is attached to test equipment (see UUT).

ECMA: European Computer Manufacturers Association.

EIA: Electronic Industries Association.

EISA: Extended Industry Standard Architecture.

Escon: Enterprise Systems Connection.

ESD: electrostatic discharge.

ESDI: Enhanced Small Disk Interface. This is the analog to SCSI. This interface, which was developed by Maxtor, provides a fast channel for disk drives. Originally, the "D" meant device, but the developers changed their aspirations and decided that a fast interface for disks was good enough. IBM did choose ESDI for early PS/2 models but is now moving exclusively to SCSI. This is one of the many interfaces that I. Dal Allan of ENDL Consulting, Saratoga, Calif., can take credit for.

FAT: file allocation table, a table that contains mappings of the physical locations of all of the clusters in all files on disk storage.

FBA: fixed-block architecture.

FC: fiber channel. This is technically an initialism and is a bad choice—writing out fiber channel makes more sense than using the initials.

fci: flux changes per inch.

FDDI: Fiber Distributed Data Interface, protocol description for optical networks and copper coaxial cable networks.

FEU: functional equivalent unit, a test unit equivalent to the actual finished device. As such, it has the same appearance as the finished article and operates in accordance with the functional requirements.

FFS: flat-file system.

FOM: figure of merit, another poorly thought-out use of acronyms. A figure of merit is a number that is determined from some weighting system that has preestablished boundaries.

fps: frames per second.

FPT: Forced Perfect Termination. This is a nifty trick developed by IBM to create a termination scheme that follows the changes in the impedance of a line. Theoretically, it implies an infinite length without attenuation of the signal. However, physics does get in the way, and the signal will die at some point.

FRAM: ferromagnetic memory.

Gbsi: gigabits per square inch.

HBA: host bus adapter. (This is SCSI talk.)

HIPPI: High-Performance Parallel Interface.

HIPPI-FP: Framing protocol.

HIPPI-LE: IEEE Std 802.2 link encapsulation.

HIPPI-PH: physical layer.

HSC: hierarchical storage controller.

Glossary *(continued)*

IDC: insulation displacement connectors.

IEC: International Electrotechnical Commission.

IEEE: Institute of Electrical and Electronics Engineers.

IIST: Institute for Information Storage Technology.

IOPS: input/output (requests) per second.

IPI: Intelligent Peripheral Interface, an interconnection standard that grew out of the Intelligent System Interface developed by ISS Sperry Univac. This interface is designed for large systems that have very high speed I/O channels.

JISC: Japan Industrial Standards Commission.

JTC1: Joint Technical Committee 1.

Lun: logical unit.

MCAV: modified constant angular velocity, a method used on compact disks and some read/write optical disks.

Mflops: millions of floating-point operations per second, a metric used to benchmark the overall processing capability of a microprocessor and math coprocessor.

MIG: metal-in-gap, a specialized read/write transducer used in high-performance disk drives.

MIPS: millions of instructions per second, a measure of a processor's horsepower. It is usually compared with a known system such as a Digital Equipment Corp. PDP 11/780 that has a MIPS rating of 1.

MO: magneto-optical, a technique of combining magnetic recording with optical recording to produce a high-capacity read/write device. Geoffrey Bate, a Santa Clara University fellow, pioneered this technique while at Verbatim Corp.

MR: magneto-resistive, a single-pole read/write transducer used in high-performance and high-bit density disk drives.

MSC: Microcomputer Standards Committee, the governing body for bus standards.

MTBF: mean time between failures.

MTBS: mean time between stops. This has nothing to do with the rapid transit system in your town; rather, it is a measure used with rotating memory systems.

MTDL: mean time to data loss. This is my favorite acronym. Apparently, it describes what we all fear most (by the way, I saved my work at this point). That's the loss of all the work we have done up to the deadline because of a power failure, your cubical mate spills coffee down the grill work of your terminal, or the gnomes who control these things have decided its your turn in the barrel.

MTSR: mean time to service repair.

MTTR: mean time to repair.

NIC: newly industrialized country. This just shows how silly we get with acronyms, but it is actually used.

NRE: nonrecurring engineering.

NRZ: nonreturn to zero, a recording method that uses the zero crossing as a reference point.

NSIC: National Storage Industry Consortium.

NWI: new work items.

OS, O/S: operating system.

OSF: Open Systems Foundation.

PCD: printed circuit disk.

POH: power-on hours.

PRML: partial response maximum likelihood.

Prej: port reject.

RISC: reduced instruction-set computer, a fast processor that has fewer instructions than a CISC and is defined as performing a register-to-memory move in one *t* (machine-cycle time).

QIC: quarter-inch cartridge; also the name of the tape committee, chaired by Raymond Freeman, Freeman Assoc., Santa Barbara, Calif., which developed the quarter-inch format and recording standards.

Rej: reject. Most people believe the general use of this term comes from the reject switch on phonograph record players that were marked "Rej" on the top of the knob.

RLL: run-length limited, which defines a code combining 1s and 0s that is a mathematical permutation allowing more information to be encoded on one transition. Notably, RLL is used in data recording and transmission.

R/W: read/write.

SCSI: Small Computer Systems Interface, one of the most important interfaces in the industry. SCSI grew out of the Shugart Associates System Interface (SASI). The now-defunct Shugart took the idea to ANSI under the guidance of I. Dal Allan (the father of modern-day interfaces), and the result was SCSI-I. Currently, SCSI-II—a robust 16-bit version with 32-bit capability—is expected to be deemed an ANSI standard soon.

SD3: project approval request (ANSI).

SPARC: Standards Planning and Requirements Committee, the group you address when preparing a new standard; also a microprocessor architecture.

SPOOL: simultaneous peripheral operations on line, when a high-speed device like a disk is interposed between a running program and a low-speed device such as a printer.

SSD: solid-state disk.

SWG: Special working group.

SSWG: Specific subject working group.

TAG: Technical Advisory Group.

TC: technical committee.

TCMM: Technical Committee on Microcomputers and Microprocessors.

TFH: thin-film head, a special read/write transducer that is made using semiconductor techniques. This type of head permits smaller geometries, thus more tracks per inch and greater

Glossary *(continued)*

density of storage.

TFM: thin-film media, a special recording media developed by using a sputtering process to lay down the thinnest possible layer of recording material to improve permeability and susceptibility to the recording process. This technique is used for optical recording, especially thermal-magneto recording (TMR) that uses both optical and magnetic methods.

tpi: tracks per inch.

TPS: transaction processing system.

UART: Universal Asynchronous Receiver Transmitter.

ULP: Upper Layer Protocol. You can tell this came from technologists who deal with systems and storage devices. Communications people have their own jargon, and they would tend to point out the layer in the seven-layer OSI model.

UUT: unit under test. (See DUT.) UUT is similar to a DUT depending on whom you talk to. Sometimes, UUT refers to software under test, while DUT refers primarily to hardware.

WORM: write-once, read-many, an optical device that laid the foundation for the optical systems being used today.

notice that it spells "Coors"—even their logo was the top of a beer can.

Remember NBI, the fast-rising word processor company of the seventies and early eighties? NBI stood for "nothing but initials."

And, one of the more interesting acronym histories belongs to *EDN* magazine. Most people erroneously think *EDN* stands for "Electronic Design News." About 40 years ago when *EDN* started, the name meant "Electrical Design News." However, the magazine became generically known as *EDN*, and the publishers decided that would be its official name. But H. Victor Drumm, the publisher during the seventies and eighties , felt *EDN* stood for "everything a designer needs," which shows that an acronym never outshines brilliance.

Want to send me your acronym story? Mail it to me at the address shown earlier; I'll be happy to receive it.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 180 Medium 181 High 182

Micro Law

continued from p. 6

prepare any response to your first.

It is the policy of Bit Bucket Consulting Services, Ltd. not to have its personnel enter into confidentiality relationships regarding pending litigation until the engineer and the company are sure they can appropriately support the position of the party for which the consulting/expert witness services are to be performed. I tried to convey this during our telephone conference on November 14, and thought that I did. But apparently I was not successful in properly communicating that position. I regret any confusion that may have been caused by failure of communication.

Accordingly, I have not read the materials enclosed in your letters and will not do so unless and until you advise me that I can do so without creating any mutual problems. I will study the two SIMM patents and any other documents of public record that you feel I can appropriately examine in the light of the foregoing policy of Bit Bucket Consulting Services, Ltd. Then I will attempt to reach a preliminary decision whether I feel that the patents are valid and whether I can otherwise support the basis of your claim against Toshiba and NEC as to SIMMs infringing your two patents. I will check with our company records to make sure no conflict exists and will promptly get back to you.

If there is any problem with this, please advise promptly. Thank you for your interest in having us assist you.

Sincerely yours,
Bit Bucket Consulting Services, Ltd.
By John Q. Kludge

This is not perfect, but probably will work. For one thing, because you cannot claim not to have seen the enclosures, it may be argued that you must have read them. I don't know what you can do about this, unless a lawyer or clean-room service reads and screens all your mail for you. I welcome suggested improvements from readers. In any event, those of you who want to consult or serve as expert witnesses now have an interesting new problem to worry about.

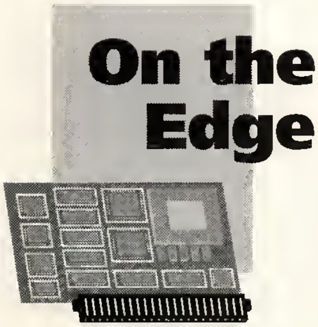
References

1. Civil Action No. 90-1477-A, E.D. Va., May 9, 1991, reprinted at 13 Comput. L. Rep. 1192 (Aug. 1991).

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 177 Medium 178 High 179



Carl Warren

McDonnell Douglas

Space Systems Company

(714) 896-3311

x. 7-1230

warren@ssdunx.mdc.com

Firmware standards

[The P1275 working group is developing a proposed standard for boot firmware based on the machine-independent Open Boot firmware. Mitch Bradley discusses Open Boot's design and benefits of standardization.

I invite readers to send information on a tool or method that solves problems, for consideration in future columns. –C.W.]

Mitch Bradley

Sun Microsystems

Firmware is the ROM-based software that controls a computer between the time it is turned on and the time the primary operating system takes control of the machine. Firmware's responsibilities include testing and initializing the hardware, determining the hardware configuration, loading (or booting) the operating system, and providing interactive debugging facilities in case of faulty hardware or software.

Historically, firmware designs have been proprietary and often specific to a particular bus or instruction set architecture (ISA). This need not be the case. Firmware can be designed to be machine-independent and easily portable to different hardware. There is a strong analogy with operating systems in this respect. Prior to the advent of the portable Unix operating system in the mid-seventies, the prevailing wisdom was that operating systems must be heavily tuned to a particular computer system design and thus effectively proprietary to the vendor of that system.

Standardization

Standardizing firmware would offer several advantages to designers and users, including

- **Consistency across different systems.** It is easier to learn one firmware command set and use it across different systems from different vendors than to have to learn a different set of commands for every different system.
- **Avoiding duplication of effort.** Without a standard, every computer manufacturer must reinvent the firmware wheel. This effort is largely wasted. The availability of standard firmware offers manufacturers the option of buying proven firmware off the shelf, rather than designing and building it from scratch. Porting is almost always cheaper than writing from scratch. The existence of a standard also will raise the level at which value is added, allowing improvements to be made on top of a proven base, rather than continually redesigning the base.
- **Reducing design cycle times.** Designers would have one less thing to build for each new system architecture if standardized firmware is available.
- **Good, not half-hearted, firmware.** Since firmware is not visible to the average user and is rarely cited in marketing literature or measured in benchmarks or reviews, it is often treated as an afterthought or a necessary evil. Consequently, firmware design has not received the same level of attention as other, more visible software components. The requirements imposed on firmware by the desire to standardize it force us to take a hard look at many issues that might otherwise be swept under the rug.

0272-1732/92/0200-0073$03.00 © 1992 IEEE

Among the many firmware solutions available on different machines, Open Boot is the only one that has been proposed as a multivendor standard.

Open Boot advantages

We designed Open Boot firmware with open systems and standards in mind. We intended from the outset to port it to many different machines with widely varying bus structures, ISAs, and configurations.

Plug-in drivers. A key Open Boot feature is support for self-identifying devices. Consider a computer with an open expansion bus, such as VMEbus or SBus. An independent board vendor (that is, not a system manufacturer) of a card that plugs into the bus wants the system to recognize and use that card. In an operating system environment, this is easily accomplished. The board vendor supplies a driver on a diskette that can then be loaded onto a hard disk or installed into the operating system.

It is more difficult to support third-party devices in the firmware environment because firmware operates before the system is ready to read the disk. Since it is difficult to merge third-party drivers into existing system ROMs, it is better to store a driver in a ROM on the card for the plug-in device to which it applies. Others have taken this approach, but most existing firmware systems store the driver in ISA-dependent machine language binary code, and thus it only works on computer systems from a particular vendor.

Open Boot also uses the plug-in driver technique. But instead of storing those drivers in machine language, Open Boot uses FCode. FCode is a machine-independent, byte-coded intermediate language for the Forth programming language. It is based on a stack-oriented virtual machine that may be easily and efficiently implemented into any computer. FCode drivers are incrementally compiled into system RAM for later execution.

In addition to its use for firmware device drivers, FCode also provides a descriptive capability. Plug-in device cards use it to report their characteristics to the firmware and system software. Such characteristics may include the device name, model, revision level, register locations, interrupt levels, supported features, and any other identification information that makes sense for the particular device. System software may use this information to automatically configure itself for correct operation with particular devices.

Interactive debuggers. Open Boot uses the same runtime system that executes FCode drivers as the basis for an interactive Forth language interpreter. This interpreter can be used as a programmable debugger, allowing developers, users, and service personnel to isolate system problems in the event of a failure.

Flexibility. We designed Open Boot for adaptability. Its notation and structure for naming particular devices is based on a hierarchical device tree that mimics the bus configuration and physical addressing of the machine on which it is implemented. This structure applies equally well to simple, single-bus desktop machines and to back-room servers with multiple processors and complicated hierarchies of interconnected buses. We designed the name space for individual device names so that allocating names requires no central authority. Companies can design their products without appealing to a master name arbiter.

The Open Boot command language is open-ended. In addition to the standard commands that are present on all implementations, an arbitrary number of new commands may be added at any time, even by the user. Such additional commands may provide access to system-specific features or may simply be customizations for the needs and tastes of individual users.

Maintainability. Field ROM upgrades can be expensive. Open Boot provides a self-patching facility that allows many types of firmware bugs to be fixed without changing the system ROM. The same facility can be used to add additional firmware capabilities to systems in the field, without changing the ROMs.

P1275 Open Boot working group

Mitch Bradley
Chair
2732 Katrina Way
Mountain View, CA 94040
phone: (415) 961-1302
fax: (415) 962-0927
or via e-mail: mitch.bradley
@eng.sun.com

Towards standardization

Sun Microsystems began developing Open Boot in 1987. We introduced version 1 with our Sparcstation 1 machines. Version 2, introduced with Sparcstation 2, corrected a number of deficiencies (we learned from our mistakes) and added several new features. We now use Open Boot on all of our current machines; approximately 500,000 units in the field employ it.

Sun has licensed its Open Boot implementation to several other computer manufacturers. Force Computers, a leading supplier of VMEbus products, intends to use Open Boot on future products across several processor families and buses. Sun Microsystems offers the source code for an implementation of Open Boot under a licensing arrangement.

Working group. The IEEE P1275 Open Boot Working Group is developing a firmware standard based on Open Boot. The group meets monthly at various locations. Membership is open to all interested parties. For more information, contact the author (see box).

Several IEEE bus standards are making provisions for Open Boot. The Futurebus+ standard includes a mechanism for using FCode ROMs for self-

identification. The VME-D draft document mentions a similar mechanism. SBus, under consideration for IEEE standardization, uses FCode.

For a copy of the P1275 draft specification contact the working group. The specification may be used and implemented without licenses or royalties.

Mitch Bradley is a senior staff engineer at Sun Microsystems. He has designed analog and digital hardware, written Unix device drivers and other software, and been a system troubleshooter. Most recently he worked on the design, implementation, and promotion of the Open Boot firmware. He owns Bradley Forthware, a small company specializing in Forth implementations for various computers.

Bradley earned a BE in electrical engineering, computer science, and math from Vanderbilt University and an MS in electrical engineering from Stanford University. He also studied for a year at Cambridge University on a Churchill Scholarship. He is a member of the IEEE Computer Society, the Forth Interest Group, and the ANSI Forth Standards Team.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Survey Card.

Low 192 Medium 193 High 194

IEEE Micro News

Send information for inclusion in Micro News one month before cover date to Managing Editor, IEEE Micro, PO Box 3014, Los Alamitos, CA 90720-1264.

The limits of chip density

Ware Myers, Contributing Editor

Semiconductor density promises to increase for many decades to come, James D. Meindl of Rensselaer Polytechnic Institute predicted in an invited lecture to Supercomputing 91 in Albuquerque last November (J.D. Meindl, "Gigascale Integration (GSI) Technology," *Proc. Supercomputing 91*, IEEE Computer Society Press, Los Alamitos, Calif., pp. 534-538). His analysis points to what he calls "gigascale integration," or one billion transistors on a chip, by about the year 2000 with continued progress for some 30 years into the new century.

Meindl analyzed a hierarchy of limits on transistor density: fundamental limits set by the laws of physics; material limits set by the physical structure and chemical composition of the likely materials, silicon and gallium arsenide; device limits such as, in the case of silicon MOSFET devices, channel length, gate oxide thickness, and others; circuit limits such as the constraints on switching logic circuits; and system limits such as clock skew.

In addition, he considered the practical limits influenced by manufacturing technology and economics. Meindl asked, "How many components/transistors can we expect to fabricate in a single silicon chip that will prove to be useful, i.e., be economically viable, at some designated future time?" He quantified the practical limits in terms of three macrovariables: minimum feature size, the square root of the die area, and the packing efficiency, that is, the number of transistors or components per minimum feature area.

From principles of physics such as the Heisenberg uncertainty principle, Meindl derived two fundamental limits which he plotted on a log-log power-time delay field. One side of these curves constitutes an impossible region, an area where the power or the time delay is less than that required to perform a switching operation.

In a similar manner he derived various other limits. He plotted limits on switching operations on the power-time delay field. Limits on transmission operations were plotted on a field of interconnection path length versus the corresponding signal propagation or response time. These lines gradually boxed in areas on the fields where operations are feasible.

Along the way he discovered that "on the basis of bulk material limits per se, silicon is not inferior to gallium arsenide as a material for gigascale integration." That is, as switching circuits get closer to the limits he hypothesizes, silicon will reassert itself over gallium arsenide's present electron-mobility advantage.

After considering a number of metrics and limits, Meindl felt that one figure of merit, in particular, was singularly instructive. The chip performance index, as he calls it, is the number of transistors on a chip, divided by the power- delay product. By power, he means the average power consumption during a binary switching transition of a logic gate. By delay,

0272-1732/92/0200-0075$03.00 © 1992 IEEE

he refers to the corresponding transition (or delay) time. Thus, as the number of transistors increases, the chip performance index also increases. And, as the power and/or delay time (in the denominator of the equation) becomes smaller, the chip performance index further increases.

This index increased by about 10^{13} from 1960 through 1990, Meindl calculated. He projects it to increase by another factor of 10^{6} from 1990 through 2020.

Tiny lasers

Researchers at AT&T Bell Laboratories have made what they believe is the world's smallest semiconductor laser: about 5 microns in diameter. Seen through a scanning electron microscope, the lasers look like microscopic thumbtacks with the head of each tack 400 atoms thick.

The lasers operate in what is called a "whispering gallery" mode, so named after the sound effect noted in large cathedrals, where a whisper along the wall can be heard all along the inside perimeter. Like these whispers, photons travel with low losses around the edge of the laser. The lasers can be used as surface- or side-emitting devices. Each semiconductor disk laser is made of one or more layers of indium gallium arsenide sandwiched between two layers of indium gallium arsenide phosphide.

AT&T's micro disk laser

Nanotechnology

As technology becomes feasible on smaller and smaller scales, scientists continue to explore organic and inorganic paths in search of building blocks for what may one day be computers that function at the molecular or atomic levels.

Natural nanocircuits. One biophysicist is studying proteins that conduct one-electron currents to find a model for protein-based nanocircuits—circuits made of organic materials 1,000 times smaller than those used today.

Jose Nelson Onuchic, an assistant professor of physics at the University of California, San Diego, believes it may be possible to design synthetic proteins that modulate minute currents, based on those found in vision, photosynthesis, and other biological functions. Onuchic believes an advantage to this line of study, as opposed to the traditional study of inorganic materials, is that nature has already evolved special proteins to perform the tasks computer designers are trying to emulate.

The goal of his research is the *biochip*, a molecular electron device that would include the protein or other organic chemical equivalents not only of wires but also of other computational gear, such as junctions, switches, resistors, and amplifiers.

Towards that goal, Onuchic and his team are pursuing three lines of research. The first is biochemical, in which researchers study the features of natural proteins that control the rate of electron tunneling within and between the amino acids that form the links of protein chains. The electron transfer rate depends on the nature of the chemical bonds traversed by the tunneling electrons as they jump from atom to atom and by the geometry of the protein. Therefore, a second line of research explores the rules that govern the folding of natural proteins into their effective shapes.

The third line of research, which builds on the results of the other two lines, is directed toward making a working molecular electronic device. These efforts aim at creating a one-molecule memory element called a shift register, along with a one-molecule amplifier to read the information stored in the shift register. Onuchic believes it might also be possible for proteins used as memory and computing elements to self-assemble on biochip surfaces, as proteins do in living cells.

Molecular manufacturing. Alternatively, some researchers are working towards molecular manufacturing, the ability to build nanomachines one molecule or atom at a time. In contrast to contemporary microcircuits, in which billions of electrons flow from one place to another, nanometer-scale circuits might perform the same operation with the change in shape or position of a single molecule.

Micro bits

Amador Corporation, a Minnesota-based, conformity assessment testing laboratory, has announced a cooperative arrangement with the German VDE Testing and Certification Institute to test **US information technology equipment** for export to Germany.

IBM and the Center for Advanced Research in Biotechnology are developing a portable software system for **computational structural biology**, including programs to compute protein structural data and model protein molecules.

Symbmath, an expert system that solves mathematic problems in symbolic formula or through numeric computation is available as shareware under the file name SM13A.ZIP from the directory MSDOS.Calculator in Simtel 20 at File Transfer Protocol sites. Symbmath requires significantly less RAM than most comparable software—640 Kbytes, as opposed to as much as 4 Mbytes.

To build such small devices, scientists must learn how to manipulate individual molecules or atoms. Current methods of molecular manipulation rely on the random motions of atoms to form the desired compounds. But molecular nanotechnology aims at moving individual atoms or molecules and snapping them into place.

Some progress has already been made. Scanning probe microscopes drag an ultra-fine stylus over a microscopic surface, thus mapping out its shape and allowing scientists to see individual atoms. Researchers have found that they can sometimes get atoms to stick to the microscope tip, enabling them to move the atoms around.

Using this technique, IBM researchers in San Jose in 1989 positioned 35 atoms of xenon to spell out their corporate logo. The achievement prompted others to create their own *nanoart*: scientists at Hitachi's Central Research Laboratory in Japan spelled out "Peace '91" by removing sulfur atoms from molybdenum disulfide, and a Stanford University student inscribed the first page of *A Tale of Two Cities* on a surface about the size of a red blood cell.

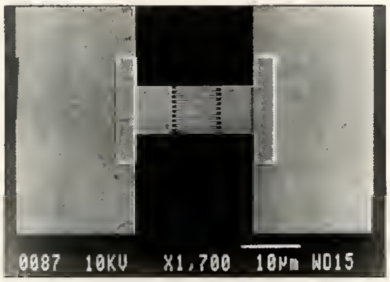
K. Eric Drexler, a visiting scholar at Stanford, is one of the founders of the Foresight Institute and the Institute of Molecular Manufacturing. He argues that manufacture at the molecular level is feasible and will one day produce nanomachines controlled by submicron, 1,000-MIPS CPUs and powered by nanomotors. Among the uses of such devices could be swimming through blood vessels to find and destroy viruses or bonding chlorine atoms from the atmosphere to protect the ozone layer. The machines would be manufactured by other nanomachines called molecular assemblers, tiny robot arms that position molecules precisely, bonding them into place in the design. (See related story on p. 7)

Alternative methods rely on the tendency of certain molecular components to stick to one another in the design of structures, thus enabling self-assembling or self-replicating devices. Nadian Seeman, a chemistry professor at New York University, has created a design that allowed DNA strands to assemble themselves into a cubelike structure. Julius Reebek, Jr., a chemist at Massachusetts Institute of Technology, designed a synthetic molecule that serves as a template for two other molecules that combine to form a copy of the original.

1.2-ps photodetectors

A graduate student at the University of Michigan has used low-temperature-grown gallium arsenide to produce what she says may be the fastest light-detecting microchip. The device's 1.2-ps response time is six times faster than commercial photodetectors.

The chip's developer, Yi Chen, credits the special type of gallium arsenide, developed by researchers at MIT, for its speed. The material responds very quickly to light, creating a current that stops and starts instantly as a laser pulse starts and stops.



Chen's interdigitated electrodes

Chen's biggest technical challenge was developing an electrode structure to match the sensitivity of the gallium arsenide. She developed a way to use submicron electron-beam lithography to create interdigitated electrodes about 0.2 microns or 2,000 angstroms apart. By condensing the photodetector's electrodes into a smaller area, the design prevents the loss of signal-carrying electrons in the gaps between electrodes. The detector achieves an internal quantum efficiency of 68 percent (collecting 68 out of every 100 electrons) as opposed to traditional electrode structures that achieve about 1 percent.

According to Chen, the device holds promise for fiber-optic communication networks hundreds of times faster than current systems, precision 3D inspection systems, vehicle collision avoidance systems, and advanced medical imaging techniques. Chen is a graduate student in the school's applied physics program and a post-doctoral fellow at AT&T Bell Laboratories.

Cobol coinventor dies

Rear Admiral Grace Murray Hopper, known as "the first lady of software," died at her home in Arlington, Virginia, on January 4. She was 85.

Hopper was a pioneer computer programmer for the US Navy and coinventor of Cobol. She is credited with coining the term *bug* to describe the problems that plague computers and programs.

Admirers described Hopper as a vigorous, tireless, and occasionally contrary woman, with a healthy contempt for those unwilling to try new ideas. She once said, "The only phrase I've ever disliked is, 'We've always done it that way.' "

Hopper earned a PhD in mathematics from Yale University and taught at Vassar College before joining the Naval Reserve in 1943. After World War II she remained in the Naval Reserve and joined the company building the Univac I. The company later merged into Sperry, where she developed an idea that lead to Cobol. She was the US's oldest active duty military officer when she retired in 1986.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 198 Medium 199 High 200

New Products

Send announcements of new microcomputer and microprocessor products to Managing Editor, IEEE Micro, *PO Box 3014, Los Alamitos, CA 90720-1264.*

Joe Hootman

University of North Dakota

Windows software

Speech from Windows

Using synthesized speech, Monologue for Windows reads aloud text, numbers, and data from Windows 3.0 and its applications. As in the manufacturer's original version, Monologue for Windows converts digital information into simulated speech in two phases using a set of phonetic translation and pronunciation rules. The software analyzes and translates text into sound descriptors, a phonetic language containing more than 1,000 rules to incorporate pitch, duration, and amplitude codes. Then it converts the language into speech signals. Algorithms drive speech tables that incorporate rules for merging signals into continuous speech.

Users can adjust the speed, pitch, and volume on screen. A dictionary manager lets users instruct the software how to pronounce words, abbreviations, acronyms, and symbols that may not comply with phonetic rules. The software requires 2 Mbytes of RAM, MS/PC-DOS 3.1 or higher, Windows 3.0 or higher, and a hard disk with at least 2 Mbytes available. *First Byte; $149.*

Reader Service No. 10

Video window

X.TV is a video window that users can manipulate like any other window on workstations running X windows. Users can reposition X.TV anywhere on the screen, scale it to full screen, or reduce it to icon size. An on-screen control panel accesses audio-video functions, including volume, brightness, and contrast. A software push-button activates the frame grabber. Images can be named and saved to a disk file in raw data, TIFF, or Targa formats.

X.TV receives images through VMEbus, SCSI, or RS-232 port buses from the manufacturer's RGB/View hardware. RGB/View processes images independently from the CPU and receives from a variety of sources including video cameras, scanning electron microscopes, and infrared devices. *RGB; $750 (X.TV), from $8,995 (RGB/View).*

Reader Service No. 11

RGB's X.TV video window

VMEbus development kit

Programmers can develop Microsoft Windows 3.0 applications in a VMEbus environment using the XVME-984 Windows VMEbus Toolkit. The kit is a utilities package designed to run with the manufacturer's line of PC/AT VMEbus processors and support modules. It includes VMEbus Manager, a Windows 3.0 application that accesses and monitors the VMEbus. A library of functions contains the low-level code that configures and communicates with the manufacturer's VME I/O products. Demonstration programs are also included that offer examples of library uses. *Xycom; $650 (with Windows), $500 (toolkit only).*

Reader Service No. 12

PC-mainframe software

The Extra PC-to-mainframe software for Mi-

0272-1732/92/0200-0078$03.00 © 1992 IEEE

Parallel-processing DSP

Texas Instruments says its TMS320C40 is the first digital signal processor built specifically for parallel processing. One of the biggest challenges facing the chip's design team was how to debug a chain or array of multiple processors. Chief architect and program manager Ray Simar had seen customers purchase individual debuggers for each processor in a parallel system.

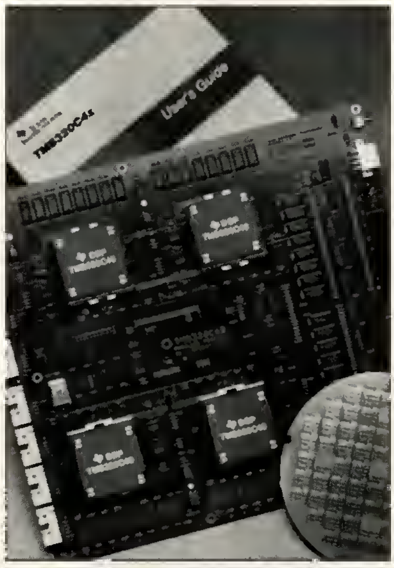
"Those debuggers cost $10,000 to $20,000 each," Simar said. "This gets prohibitive if you have 16 processors."

Space becomes a big problem, too. "They'd have nowhere to put them," he added, "so they'd have them hanging from the ceilings."

TI avoids rooms full of hanging debugger boxes by incorporating an analysis module onto each C40 chip. The module links through a JTAG (IEEE Std. 1149.1) port to debugger software, accessible in a windows-format interface. Users can monitor and test processors in the system individually or globally.

The analysis module is one of the key features of the C40, which Simar says is the first DSP built specifically for parallel processing. His team designed the chip from the ground up after looking at customers' difficulties incorporating their previous-generation DSP, the C30, into parallel processing systems.

According to Simar, customers linked the C30s together with multi-



Texas Instruments' TMS320C40 DSP

chip FIFO memories, an expensive process that also takes up space. "What we've done, essentially, is integrate 12 FIFOs onto the chip," he said. The C40 has six communication ports that link directly to other C40s with no external logic required. Thus, they link together tightly in pipelines, 2D arrays, or 3D arrays. This close proximity helps the C40 achieve what TI says is the highest performance of any floating-point processor: 275 MOPs with 320-Mbyte/s throughput per C40. Simar says the chips achieve even higher performance in parallel processing systems.

An on-chip, six-channel DMA coprocessor acts as a clerk for the on-chip CPU. To allow the CPU to function smoothly, the DMA receives data, assembles it into packets, and stores it in a memory buffer (if necessary) until the CPU is ready for it. The C40 has an 8-Kbyte memory and two external memory buses to global and local memory. Through the interface, users choose how much on-chip memory space is allocated for the DMA and how much for the CPU.

Among the tools available from the manufacturer are an in-circuit emulator that provides parallel debug capabilities for embedded applications, a parallel-processing development system, an ANSI-compatible C compiler with parallel-processing runtime support library, an assembler and linker, and a state-accurate simulator. Third-party tools include the Multiprox code generation systems, an optimizing Ada compiler, and the Spox real-time operating system.

Simar says the C40 is suited for medical imaging (ultrasound, magnetic resonance, and CAT scans), pattern recognition for robotics, radar processing, 3D graphics, and military uses (image recognition and tracking). He also sees prospects for some nontraditional uses, including high-performance accelerators that attach directly to personal computers or workstations, high-bandwidth data transmission, and high-speed network interface. TI plans to ship in volume by mid 1992. *Texas Instruments.*

Reader Service No. 13

crosoft Windows includes several easy-to-use features. Dynamic data exchange macros create automated information links between the mainframe and Windows applications. APL character support assists in financial and statistical analysis.

Extra also supports light pens, a feature designed primarily for the health care industry. Command-line file transfer capability, for users more comfortable with DOS syntax, allows concurrent transfer of files using macros. A diagnostic trace facility allows users to record communications events on the network, to identify problems. *Attachmate; $425, $75 (upgrades).*

Reader Service No. 14

Cut and paste from X to MS

An X server integrates the X Window System in a Microsoft Windows environment. Xoftware for Windows allows users to cut and paste between X and MS Windows applications and use MS Windows as a local window manager. It also features a complete on-line help system and several conveniences for

installing, configuring, and starting X applications from within the MS Windows environment.

Xoftware for PC Unix, release 2.1, is an X server for 386/486-based machines running SCO or Interactive Unix. It adds support for Texas Instruments' 34020-based graphics accelerator board and a hot-key function allows users to switch to SCOs Multiscreen application. *Age; $495 (Xoftware for Windows), $595 (Xoftware for PC Unix).*

Reader Service No. 15

DOS-to-Unix connectivity

Aterm software allows two-way file transfer between systems and can emulate ANSI color terminals, graphics support, multiple screen display, and a menu-driven interface. It features a hot-key function that allows seamless movement between DOS and Unix systems. PC users can run DOS applications and switch immediately to applications running on the Unix host. Aterm is compatible with MS-DOS and Windows. *Specialix; from $125.*

Reader Service No. 16

Signal-processing hardware and software

Background data collection

Easy Data collects data, modifies it, and sends it to the keyboard buffer without affecting normal keyboard functions. It imports data in the background while users work with another program in the foreground. Easy Data works with most software that receives data entry through the keyboard. Keyboard characters and macros can be inserted automatically before and after each data field to simulate the same keys that would be pressed in manual data entry. Data can also be selectively parsed so that only the required data transfers. *Labtronics; $145.*

Reader Service No. 17

1,280-Mflops performance

Two of Texas Instruments' C40 DSP chips (see box on p. 79 and diagram below) are put to use in the Spirit-40 AT, an 80-Mflops DSP engine. Each 40-Mflops C40 includes 1 Mbyte of SRAM on the main bus, 1 Mbyte of SRAM on the local bus, 64 Kbytes of boot EPROM, and memory expansion for up to 16 Mbytes of DRAM.

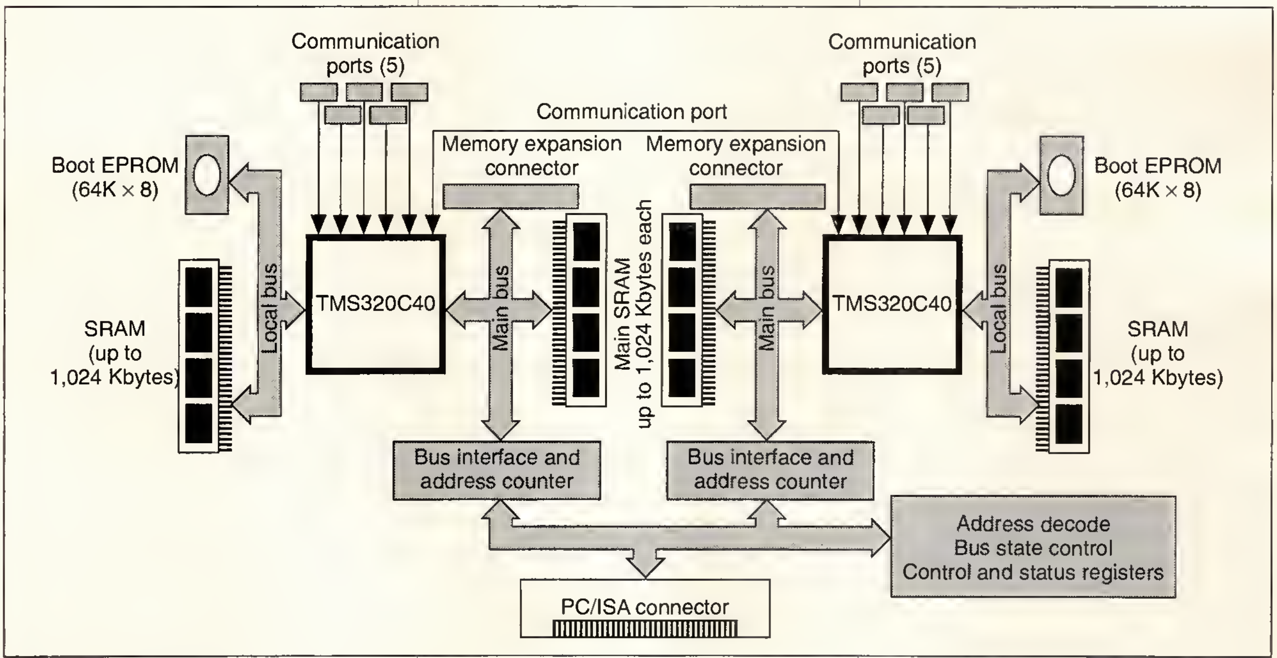
With the C40's six communication links, the Spirit-40 configures up to a six-dimensional hypercube engine with 64 nodes. The board occupies a 16-bit ISA slot and is compatible with 386- and 486-based machines. Up to eight boards fit in a passive backplane ISA system, yielding 640-Mflops performance. Two backplanes yield 1,280-Mflops peak performance. *Sonitech International; $8,995.*

Reader Service No. 18

DSP package

Filter designers can analyze the performance of their designs on Monarch, a menu-driven DSP package with filter design, signal analysis, and graphical capabilities. Users can design finite impulse response filters (up to a maximum order of 512) or infinite impulse response filters.

The package includes Siglab, a DSP language with over 100 mathematical and system operations for performing



Sonitech's Spirit-40 AT

READER SERVICE CARD

IEEE MICRO

February 1992 issue (card void after August 1992)

Name ______________________

Title ______________________

Company ______________________

Address ______________________

City ______________ State ______ Zip ________

Country ______________ Phone(______) ________

Reader Interest
(Add comments on the back)

Readers,
Indicate your interest in articles and departments **by circling the appropriate number** (shown on the last page of articles and departments):

150	151	152	180	181	182
153	154	155	183	184	185
156	157	158	186	187	188
159	160	161	189	190	191
162	163	164	192	193	194
165	166	167	195	196	197
168	169	170	198	199	200
171	172	173	201	202	203
174	175	176	204	205	206
177	178	179	207	208	209

Product Information
(Circle the numbers to receive product information)

1	21	41	61	81	101	121	141
2	22	42	62	82	102	122	142
3	23	43	63	83	103	123	143
4	24	44	64	84	104	124	144
5	25	45	65	85	105	125	145
6	26	46	66	86	106	126	146
7	27	47	67	87	107	127	147
8	28	48	68	88	108	128	148
9	29	49	69	89	109	129	149
10	30	50	70	90	110	130	
11	31	51	71	91	111	131	
12	32	52	72	92	112	132	
13	33	53	73	93	113	133	
14	34	54	74	94	114	134	
15	35	55	75	95	115	135	
16	36	56	76	96	116	136	
17	37	57	77	97	117	137	
18	38	58	78	98	118	138	
19	39	59	79	99	119	139	
20	40	60	80	100	120	140	

LIBRARY ORDER CARD

IEEE MICRO

Does your library subscribe?

Readers: To recommend *IEEE Micro* for acquisition, complete this card and submit it to your librarian or department head.

Attention: Librarian/Department Head
I have examined *IEEE Micro* and would like to recommend the magazine for acquisition.

Name (please print) ______________________

Department ______________________

Date ______________________

Signature ______________________

Sample copies are available from:

IEEE Computer Society
PO Box 3014
10662 Los Vaqueros Circle
Los Alamitos, CA 90720-1264

IEEE Micro
ISSN 0272-1732
Bimonthly: $157/yr.

SUBSCRIPTION CARD

SUBSCRIBE TO IEEE MICRO

All the facts about today's chips and systems

❑ **YES**, sign me up!

If you are a member of the Computer Society or any other IEEE society, pay the member rate of only $23 for a year's subscription (six issues).

Society: ______________________

IEEE membership no: ______________________

Society members: Subscriptions are annualized. For orders submitted March through August, pay half the full-year rate ($11.50) for three bimonthly issues.

❑ **YES**, sign me up!

If you are a member of ACM, ACS, BCS, IEE (UK), IEEE but not a member of an IEEE society), IECEJ, IPSJ, NSPE, SCS, or other professional society, pay the sister-society rate of only $39 for a year's subscription (six issues).

Organization: ______________ Membership no: ______________

Full Signature ______________ Date

Name ______________________

Street ______________________

City ______________________

State/Country ______________ ZIP/Postal Code

❑ Payment enclosed *Residents of CA, DC, Canada, and Belgium add applicable tax.*

❑ Charge to ❑ Visa ❑ MasterCard ❑ American Express

Charge-card number

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Expiration date

_ _ _ _ Month _ _ _ _ Year

Charge orders also taken by phone:
(714) 821-8380 8 a.m. to 5 p.m. Pacific time
Circulation Dept.
10662 Los Vaqueros Circ., PO Box 3014
Los Alamitos, CA 90720-1264

Prices valid through 12/31/92
02/92 MICRO

Editorial comments

I liked: ______________________________

____

____

I disliked: ______________________________

____

____

I would like to see: ______________________________

____

____

Reviewers Needed. *If interested, send professional data to Dante Del Corso, Dipartimento di Elettronica, Politecnico di Torino, C.so Duca degli Abruzzi, 24, 10129, Torino, Italy.*

For reader service inquiries, see other side.

PO Box is for reader service cards only.

PLACE POSTAGE HERE

IEEE Micro
PO BOX 16508
NORTH HOLLYWOOD CA 91615-6508
USA

IEEE MICRO

IEEE Micro addresses an international audience of professionals who design and use microprocessors and microcomputers. In-depth articles examine microprocessor and board-level design, applications, system integration, VLSI, ASICs, fault tolerance, and hardware/software environments. Departments cover legal issues; industry and technical news from the US, Europe, and Japan; new products; standards; tools; and book and software reviews. *IEEE Micro* gives readers the information they need to stay competitive in this ever-expanding field.

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 38 LOS ALAMITOS, CA

POSTAGE WILL BE PAID BY ADDRESSEE

IEEE COMPUTER SOCIETY
PO BOX 3014
LOS ALAMITOS CA 90720-9804
USA

signal and system analysis. Designers can create their own algorithms and DSP operations, which can be synthesized, tested, and saved for future use in other applications. Graphical displays feature overlay, zoom, grid, color, line style, and data location support. Monarch simulates real-world conditions, such as noise and fixed-point analysis. It requires MS-DOS 3.0 or higher, 640-Kbyte RAM, and a hard disk (or 2-Mbyte floppy). *Dynacomp; $549.95, $10 (demonstration disk).*

Reader Service No. 19

Communication hardware and software

Controls 255 nodes

COM20010 is a token-passing communications controller designed for high-speed intelligent data highways. The 2.5-Mbps device features a microcontroller interface and 1K × 8 onboard buffer RAM. It uses an Arcnet protocol engine and a token-passing protocol for real-time deterministic performance that supports data packet sizes from 4 to 512 bytes.

Up to 255 peer-to-peer nodes can network via the COM20010 at a data rate of 2.5 Mbps. System designers can implement peer-to-peer or master-slave communication schemes. The CMOS-fabricated device runs on a +5V power supply and comes in commercial and industrial temperature ranges. *Standard Microsystems; $11.36 (1,000s).*

Reader Service No. 20

LANs connect to WANs

Local area networks connect into wide area networks with the LAN Transport Management System. LAN TMS, suited for Token Ring and Ethernet environments, eases interconnection between diverse LAN protocols, such as IBM Net BIOS, SNA, TCP/IP, Novell Netware, IBM server, and Banyan Vines. Synchronous data link control pass-through capability enables it to merge a non-LAN serial data stream with regular Token Ring traffic for transmission over a common WAN link. Network administrators can monitor and troubleshoot problems on geographically dispersed LANs from a central management console. *General Datacomm; from $4,600.*

Reader Service No. 21

Gateway software

Up to 30 Netware users can access transmission control protocol and internal protocol applications with Catipult gateway software. Its TCP/IP applications run over IPX, using Novell's Net BIOS. From the user's PC, Catipult's applications are routed to the OS/2 gateway, which replaces IPX with the TCP/IP transport protocols and sends the application packets to other hosts. The product runs on OS/2 PCs equipped with a network interface card supported by Novell's Netware Requester for OS/2 and over Ethernet, Token Ring, Arcnet, and Broadband networks running Netware 2.x or Netware 3.x. *Ipswitch; $2,975.*

Reader Service No. 22

Sparcstations link to IBM

Two software products connect Sun Microsystems Sparcstations and IBM's system network architecture (SNA) over a Token Ring. Brx TR/SNA transparently supports all SNA services and supports local network management. Brx TR/IP leverages the transfer functions of the Token Ring network to include Unix systems. The technology enables traditional TCP/IP applications such as NFS, File Transfer, Mail, and Remote Login to operate seamlessly over Token Ring. Users can access remote Sparcstations as if they were locally connected to the same network. *Brixton Systems; $995 (both products and Token Ring card).*

Reader Service No. 23

Special boards

Video interface card

The MZ-UT01 interface card establishes a video rate path between the Scorpion real-time VGA frame grabber and Eighteen Eight Laboratories' PL2500 floating-point array processors. Multiple 640 × 480 × 8-bit images can be transferred in real time between the video card and the array processor. The half-length mezzanine card mounts directly on the PL2500 and draws power and ground through Span 32 bus interface connectors. The MZ-UT01 card comes with software to control the Scorpion board, interconnection cables, and a reference manual. *Univision Technologies; $2,495.*

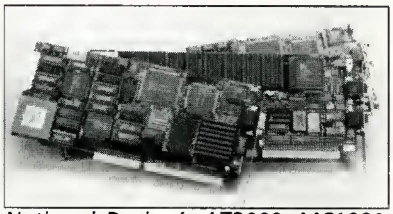
Reader Service No. 24

16.7 million colors

The Volante family of graphics processors offers three alternatives to PC users. The AT2000, compatible with PC ATs, displays up to 16.7 million colors at 640 × 480 resolution. At higher resolution (up to 1,024 × 768) it displays 32,768 colors. The MC1000, compatible with Micro Channel PCs, displays 256 colors at 1,024 × 768 resolution. The AT800, also compatible with PC ATs, displays 256 colors.

The processors, based on Texas Instruments' TMS34020, have a refresh rate of 72 Hz and a maximum bandwidth of 73 MHz. Their graphical interface works with MS Windows, Tiga, X Windows, and Nova Graphics CGI. *National Design; $1,295 (AT2000), $995 (MC1000), $795 (AT800).*

Reader Service No. 25



National Design's AT2000, MC1000, AT800

Flicker-free screens

The Windows VGA 8800 graphics board boosts processing speeds for Windows 3.0 menus, windows, scroll-

continued on p. 83

64-bit RISC microprocessor

Mips has extended its RISC architecture to the 64-bit format with the R4000 microprocessor, a 1.3-million transistor chip designed to handle the growing volume of data found not only in large processors but in desktop machines as well.

According to Andy Keane, product development manager for the R4000, company designers looked in the direction RISC architecture was moving and decided to target the volume desktop market. "The early focus (of RISC) was on large machines, especially for databases," Keane said. "Now, RISC is more broadly defined." The R4000 is aimed at a range of applications, including high-end PCs, graphics workstations, database servers, and multiprocessing systems.

As users grapple with increasingly large amounts of data, the current generation of 32-bit machines, which handles up to 4 Gbytes of information, is becoming obsolete.

Based on observations that the memory requirements of the average program grow by a factor of 1.5 to 2 each year, the company expects mainstream applications to exceed the capability of 32-bit machines by the mid-nineties. Doubling the number of bits in the microprocessor enables it to handle up to a terabyte of data.

Keane said his company is targeting PC users who are interested in the performance of a workstation but don't want to give up their investment in software. The R4000 is compatible with R3000 and R6000 applications, as well as Windows NT, Santa Cruz Operation's Desktop, and RISC/os, Mips' implementation of Unix. The chip performs in a 32-bit subset mode for most applications, employing its larger address only when necessary.

A 50-MHz clock drives the chip and operates a 100-MHz superpipeline. On-chip CPU components include a 64-bit integer processor, 64-bit floating-coprocessor, memory management unit, 8-Kbyte instruction cache, 8-Kbyte data cache, control and management facilities for primary and secondary cache, and multiprocessing capabilities.

The R4000 comes in three versions.

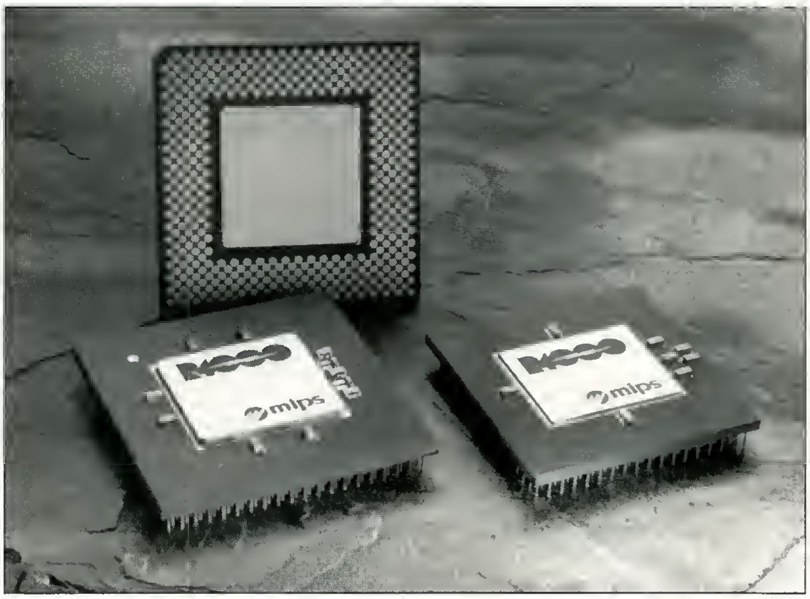
- R4000PC, which supports primary on-chip cache, is a 179-pin grid array package aimed at desktop, low-end servers, and embedded control systems.
- R4000SC, with secondary cache for uniprocessing applications, is offered in 447-PGA or land grid array packages intended for high-performance desktops and servers.
- R4000MC, with multiprocessing features and secondary cache, also comes in a 447-PGA or -LGA package.

Based on simulations of the SPEC benchmark suite, performance ranges from 40 Specmarks for the R4000PC to 60 Specmarks for the other two versions. According to the company, comparable Specmark performance has previously been achieved only from implementations of five to nine chips.

Among the development tools available are a C RISC compiler and a systems programmer package. The latter includes a cache memory simulator, an architecture simulator, and a development package.

Through the Advanced Computing Environment Initiative, a consortium of software, systems, and semiconductor companies that promotes an open computing environment, Mips has licensed six semiconductor manufacturers to produce the R4000. Last year Olivetti, Acer, and Mips previewed PC-size machines using the R4000. Mips expects commercially available machines using its microprocessor available in 1992. *Mips; $700 to $1,200, through licensed manufacturers.*

Reader Service No. 26



Mips' R4000PC, R4000SC, R4000MC

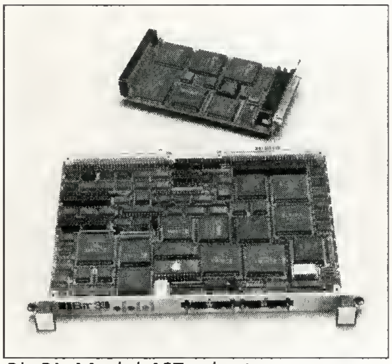
ing, and fonts up to 30 times faster than Super VGA standards, according to the manufacturer. A refresh rate of 70 Hz reduces screen flicker. Other features include 1 Mbyte of memory, 256 colors, and resolution up to 1,280 × 968 pixels. *Genoa Systems; $495.*

Reader Service No. 27

CSBus-to-VMEbus link

Sun Sparcstations can interface with a VMEbus system with the Model 467 adapter. The adapter includes an SBus card and a 6U VME card, connected with a shielded cable. The systems communicate in two ways. Memory mapping permits the SBus and VMEbus host processor in one chassis to execute random access reads and writes to the destination bus as it would to a local memory. Alternatively, a built-in DMA controller permits transfers from the memory of one system to the other as fast as 20 Mbytes/s. *Bit 3 Computer Corp.; $2,850.*

Reader Service No. 28



Bit 3's Model 467 adapter

Storage devices

Solid-state disk emulator

The Blue Flame III solid-state, nonvolatile disk emulator boosts access speed by up to 20 times. The DOS-compatible card is an I/O mapped device built from 14 SIMMs. It features a 16-bit data path and transfers data up to 4 Mbytes/s. Capacities range from 2 to 56 Mbytes. Each card fits in a full-length, 16-bit ISA bus slot. *Semi Disk Systems; from $595.*

Reader Service No. 29

20-Mbyte Mac floppy

Quad Flextra is a very high density floppy-disk subsystem for the Macintosh with a 35-ms seek time and a data transfer rate of 1.25 Mbytes/s. Each 3.5-in. disk in the subsystem has a formatted capacity of 20.4 Mbytes, enough to hold font libraries, large graphics, and desktop publishing files. The external floppy subsystem measures 2.25 × 6.81 × 8.38-in., weighs less than 4 lbs., and has two SCSI connectors. The system includes a shielded cable, driver, utility software, and four disks. *Quadram; $895, $25 (additional disks).*

Reader Service No. 30

4.3-Gbyte tape drive

Two quarter-inch cartridge tape drives store up to 2.15 Gbytes of uncompressed data or 4.3 Gbytes of compressed data. The 9200 and the 9200C use a track density of 30 serpentine recording tracks and a record density of 67,733 bpi.

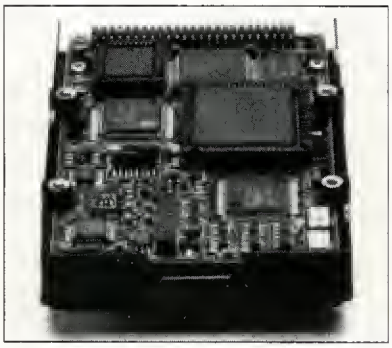
The drives support a synchronous burst transfer rate of 4.8 Mbytes/s and a sustained host transfer rate of 400 to 600 Kbytes/s for the 9200 and from 800 to 1,200 Kbytes/s for the 9200C. According to the manufacturer, each device accesses any file on a tape in two minutes or less and specifies a nonrecoverable error rate of no more than 1 in 10^{15} bits. *Wangtek; $900 (9200), $1,100 (9200C) (OEM quantities.)*

Reader Service No. 31

Winchesters for portables

Portable, laptop, and notebook computers can support more complex applications with the higher storage capacities of two Winchester disk drives. The MK-2024FC has a formatted capacity of 86 Mbytes and an average access time of 19 ms. The MK-2124FC holds 130 Mbytes and supports 17-ms access. Both drives come in a 0.75 × 2.5-in., 6-oz. package and consume 1.8 watts when active and 0.15 W when inactive. Each drive has a 32-Kbyte cache memory and a 5-Mbyte/s data transfer rate. *Toshiba; $495 (MK-2024FC) in (OEM quantities), $695 (MK-2124FC).*

Reader Service No. 32



Toshiba's MK-2024FC

Laptop upgrades

Laptop Solutions upgrades hard disks for Toshiba laptops and notebooks, to store up to 120 Mbytes of data. Each upgrade includes installation and partitioning of the hard drive to user specifications, formatting of each logical drive, complete system diagnostics, and a one-year parts and labor warranty. The Houston-based company guarantees a 48-hour turnaround, including a 24-hour burn-in and test. *Laptop Solutions; from $695.*

Reader Service No. 33

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 189 Medium 190 High 191

Product Summary

Joe Hootman
University of North Dakota

Manufacturer	Model	Comments	R.S.#
Chips			
Cirrus Logic	CL-GD6411 Graphics controller	A 3.3V, one-chip VGA controller supports prototyping and system development for notebook computers with 64 gray shades on a monochrome LCD. The device can directly drive a 512-color, active-matrix LCD. Features include simultaneous LCD and CRT displays, host bus interface logic, RAMDAC, and memory control logic. 160-pin quad flat pack; $65 (100s).	80
Cybernetic Micro Systems	CY545B/CY500 controllers	The CMOS CY545B uses pulse and direction signals to generate up to 27,000 steps/s for full-, half-, quad-, and microstep applications. The CY500 provides programmable acceleration slopes for applications requiring 1 step/minute to 2,000 steps/s. Both accept ASCII or binary commands from a serial or parallel host. 44-pin PLCC or 40-pin DIP from $25 (1,000s) (CY545); 40-pin LSI from $10 (1,000s) (CY500).	81
Linear Technology	LTC1235 supervisor	Microprocessor supervisory circuit resets at 1V and offers a conditional battery backup feature for RAM data. Available in commercial or industrial grades and 16-pin SO or plastic DIP. $3.85 (100s) (plastic, commercial).	82
LSI Logic	Sparkit-40/SS2 chip set	Manufacturers developing workstations compatible with the Sun Sparcstation 2 can sample the 40-MHz chip set, which includes the L64841 MMU, L64844 cache controller, and the L64846 DRAM controller. Manufactured for Sun and sold under license, the set works with three graphics controllers and Sunsoft software. $844 (100s).	83
Philips Semiconductors	83C528 controller	This 80C51-compatible microcontroller features a watchdog timer, 32 Kbytes of ROM, and 512 bytes of on-chip RAM. The extra RAM allows the CMOS device to run compiled application programs in PL/M and C languages and provides space for context switching for stack enhancements in internal memory. DIPs, PLCCs, or quad flat packs; EPROM and one-time-programmable versions available.	
Pletronics	Clock oscillators	Line of 50- to 120-MHz clock oscillators produces HCMOS/ACMOS-compatible output and drives 15 standard TTL loads. The crystal operates in fundamental mode at all frequencies. DIPs and mini DIPS with plastic/J and Gull-Wing leads; from $3 (1,000s).	85
Xicor	X24C00 EEPROM	A 128-bit, CMOS serial device interfaces directly to a 2-wire serial bus and features software protocol that allows operation at a 1-MHz clock rate. 8-pin plastic DIP, type P, and plastic SOIC, type S; from 45 cents (1,000s).	86

Manufacturer	Model	Comments	R.S.#
Boards			
Data Translation	DT385 image processors	One-monitor, PC AT/EISA-compatible boards combine a frame grabber and graphics processor to acquire, digitize, and display standard and nonstandard video signals on a VGA monitor. An on-board TMS34020 processor accelerates Windows 3.0 graphics display; permits scaling and variable screen placement; and speeds arithmetic, convolutions, and morphological operations. From $2,995, depending on memory.	87
Micro Express	Hi-Color Turbo VGA card	Designed around the Tseng Labs graphic chip set and the Sierra Semiconductor 15-bit DAC, the 32,768-color card supports 800 × 600 and 640 × 480 resolutions, as well as 1,024 × 768 resolution with 256 colors. A Turbo switch permits the card to switch between zero- and one-wait-state operation. $185.	88
Newer Technology	fx/Overdrive accelerator	Variable-speed (40-/55-MHz) accelerator, when running at its fastest setting, promises to speed stock Macintosh IIfx performance approximately 40 percent. When teamed with 16-Mbyte SIMMs, the surface mount device lets the IIfx act as a workstation. The accelerator's motherboard installation leaves the Nubus and PDS slots open.	89
Parsytec	BBK-S4 Sbus/ transputer interface	Adapter board and accompanying software let Sparcstations and compatibles serve as a standard host to large-scale transputer systems for data transfers up to 8.8 Mbytes/s. The T225 and controller-equipped Sbus slave also assists image processing, real-time, and other computation-intensive applications and links with up to four external systems. $3,950; quantity pricing available (10s).	90
Software			
Microware Systems	Polytron source code control	Utility package for automatic version control of application source code supports OS-9 and OS-9000 real-time operating systems. The set of 10 integrated modules supports multiuser development environments and features built-in file and user-access security. From $895; available on disk or tape.	91
MIPS Computer Systems	R3000 Riscross tools	RISC microprocessor software development tools run on Sun-4/ Sun OS workstations and VAX/VMS systems. The cross-development tools include a K&R-compliant C compiler, System Programmer's Package, Cache 3000 memory simulator, and SPP/e development package. $40,000 (set); from $9,000 (individual prices).	92
Slate Corporation	Penbook	Electronic book Author and Reader software lets users translate, compress, and store Postscript documents and read them on a pen-based computer. Documents displayed as mixed text/ graphics pages can be searched for user-defined words or phrases. A markup layer lets users annotate documents with personal notes. $695 (Author); $99 (Reader).	93

Manufacturer	Model	Comments	R.S.#
Peripherals			
HDS	Viewstation FX terminals	X Window, RISC-based, 14-, 16-, and 19-inch, 256-color terminals feature an Intel i960 CPU and two ASIC processors for communications and graphics integrated onto one board. Local Open Look and Motif window managers reduce network traffic and improve interactive performance. From $2,799.	94
Mitsubishi	P-78U video printer	Monochrome autoscanning video printer promises to deliver 6 × 8-inch, A4-size prints with 1,280 horizontal dots per line in 256 shades of gray in 24 seconds. Composite, S-VHS, analog, TTL RGB, and Centronics parallel port input print in positive/negative, mirror-image, and multiformat print styles. $2,999.	95
RGB Spectrum	1600U scan converter	Video scan converter with zoom, antialiasing, and 24-bit color processing changes high-resolution computer graphics to television format in real time. The unit automatically synchronizes to computer displays with 20/90-kHz horizontal scan rates. An optional RS-232 port lets users control all functions from a computer or ASCII terminal.	96
Miscellaneous			
Aristo Computers	Simcheck adapters	Memory tester family adds four adapters (ZIP memory chips; PLCC and SOJ memory chips; bank; and Apple Macintosh IIfx SIMM). Basic Simcheck tests standard SIMM and SIP memory modules with 8 or 9 bits of 64K to 16 Mbytes. A two-line LCD shows instructions and test results. $99 to $345 (adapters), $995 (Simcheck).	97
Lucas Duralith	LDC100 controller	Controller with internal EEPROM, 8-bit A/D converter, and serial interface lets designers test the applicability of touch-screen technology to their products by touching the screen at two opposite corners of the active display area. The controller is part of a development kit that includes a touch screen, two product development software disks, a user's guide, 220V to 110V transformer and plugs, and appropriate cables and connectors. $695.	98
Philips Semiconductors	KM110BH/1x sensors	Hybrid magnetoresistive modules measure rotational speed down to zero using a toothed tachometer wheel with an inductive or Hall-effector sensor. Separations between tooth and sensor can be several millimeters, and tooth structure does not have to be defined closely. Samples available.	99
Solectek	Pocket fax modem	Portable, 9,600/4,800-bps send/receive modem supports DOS, Windows, and Macintosh applications. Users send faxes from within applications via a pop-up fax menu; they may continue working in the foreground or leave the application by entering a telephone number and normal printing commands. From $299.95, depending on application.	100

Software Report

continued from p. 9

Micromachines will make it possible to create new medical equipment capable of performing diagnosis and treatment simply, accurately, and with less need for surgery. They will also contribute to the advancement of microsurgery techniques (eye surgery, suture of microscopic blood vessels, and so on) and the development of artificial organs to be placed inside the body. Biotechnologists will apply micromachines to cellular manipulation such as well separation, injections into cells, and cell fusion.

Announcements

NSF sponsors an active program for US scientists interested in working in Japan. As part of that program NSF prepared a directory of 150 Japanese companies that are willing to receive American researchers at their laboratories. The directory lists the company name, activity information, personnel, facilities, and research they will support, along with contact information. For copies of the directory, write to Japan Programs, Division of International Programs, National Science Foundation, Washington, DC 20550; fax (202) 357-5839.

The English summary of the outcome of the preliminary study on NIPT (or Sixth Generation Project) that I reported on in the April issue has been released. The Industrial Electronics Division of MITI published the 150-page "Report of the Research Committee on New Information Processing Technology" in March 1991.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 195 Medium 196 High 197

Coming Next Issue

The April Issue of *IEEE Micro* features articles selected from presentations at the third annual Hot Chips Symposium sponsored by the IEEE Computer Society's Technical Committee on Microprocessors and Microcomputers.

Don't miss

Hot Chips III

Read the April 1992 Issue of

IEEE MICRO

IEEE COMPUTER SOCIETY®

A member society of the
Institute of Electrical and Electronics Engineers, Inc.

THE FOLLOWING INFORMATION IS AVAILABLE:

Contact the Publications Office;
to facilitate handling, please request by number.

- Membership application, student #203, others #202
- Periodicals subscription form for individuals #200
- Periodicals subscription form for organizations #199
- Publications catalog #201
- Compmail electronic mail brochure #194
- Technical committee list/application #197
- Chapters lists, start-up procedures #193
- Student scholarship information #192
- Volunteer leaders/staff directory #196
- IEEE senior member grade application #204
  (requires ten years practice and significant performance in five of those ten)

To check membership status or report a change of address, call the IEEE toll-free number, 1-800-678-4333. Direct all other Computer Society related questions to the Publications Office.

PURPOSE

The IEEE Computer Society advances the theory and practice of computer science and engineering, promotes the exchange of technical information among 100,000 members worldwide, and provides a wide range of services to members and nonmembers.

MEMBERSHIP

Members receive the acclaimed monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.

PUBLICATIONS AND ACTIVITIES

Computer. An authoritative, easy-to-read magazine containing tutorial and in-depth articles on topics across the computer field, plus news, conferences, calendar, interviews, and product reviews.

Periodicals. The society publishes seven magazines and five research transactions. Refer to membership application or request information as noted above.

Conference Proceedings, Tutorial Texts, Standard Documents. The Computer Society Press publishes more than 100 titles every year.

Standards Working Groups. Over 100 of these groups produce IEEE standards used throughout the industrial world.

Technical Committees. More than 30 TCs publish newsletters, provide interaction with peers in specialty areas, and directly influence standards, conferences, and education.

Conferences/Education. The society holds about 100 conferences each year and sponsors many educational activities, including computing science accreditation.

Chapters. Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.

OMBUDSMAN

Members experiencing problems — magazine delivery, membership status, or unresolved complaints — may write to the ombudsman at the Publications Office.

EXECUTIVE COMMITTEE

President: Bruce D. Shriver*
17 Bethea Drive
Ossining, NY 10562
Phone: (914) 762-8030
Fax: (914) 941-9181

President-Elect: James H. Aylor*
Past President: Duncan H. Lawrie*

VP, Conferences and Tutorials: Barry W. Johnson (1st VP)*
VP, Educational Activities: Raymond E. Miller (2nd VP)*
VP, Membership Activities: Fiorenza C. Albert-Howard*
VP, Press Activities: Yale N. Patt*
VP, Publications: Harold S. Stone*
VP, Standards Activities: Gary Robinson†
VP, Technical Activities: Joseph Boykin†

Secretary: Ronald G. Hoelzeman*
Treasurer: Laurel V. Kaleda†
IEEE Division V Director: Bill D. Carroll†
IEEE Division VIII Director: Helen M. Wood†
Executive Director: T. Michael Elliott†

*voting member of the Board of Governors
†nonvoting member of the Board of Governors

BOARD OF GOVERNORS

Term Expiring 1992:
Alicja I. Ellis, Ronald G. Hoelzeman, Tadao Ichikawa, C.V. Ramamoorthy, Sallie V. Sheppard, Harold Stone, Akihiko Yamada

Term Expiring 1993:
Fiorenza Albert-Howard, Jon T. Butler, Michael C. Mulder, Yale N. Patt, Anneliese von Mayrhauser, Benjamin W. Wah, Ronald Waxman

Term Expiring 1994:
Mario R. Barbacci, Luis-Felipe Cabrera, Wolfgang K. Giloi, Guylaine M. Pollock, John P. Riganati, Ronald D. Williams, Thomas W. Williams

Next Board Meeting
February 28, 1992, 8:30 a.m.
Cathedral Hill Hotel, San Francisco, CA

SENIOR STAFF

Executive Director: T. Michael Elliott
Publisher: H. True Seaborn
Director, Conferences and Tutorials: Anne Marie Kelly
Director, Finance and Information Services: Tod S. Heisler
Director, Board and Administrative Services: Violet S. Doan
Assistant to the Executive Director: Sandra K. Pfau

COMPUTER SOCIETY OFFICES

Headquarters Office
1730 Massachusetts Ave. NW
Washington, DC 20036-1903
Phone: (202) 371-0101
Fax: (202) 728-9614

Publications Office
10662 Los Vaqueros Cir.
PO Box 3014
Los Alamitos, CA 90720-1264
Membership and General Information:
(714) 821-8380
Publication Orders: (800) 272-6657
Fax: (714) 821-4010

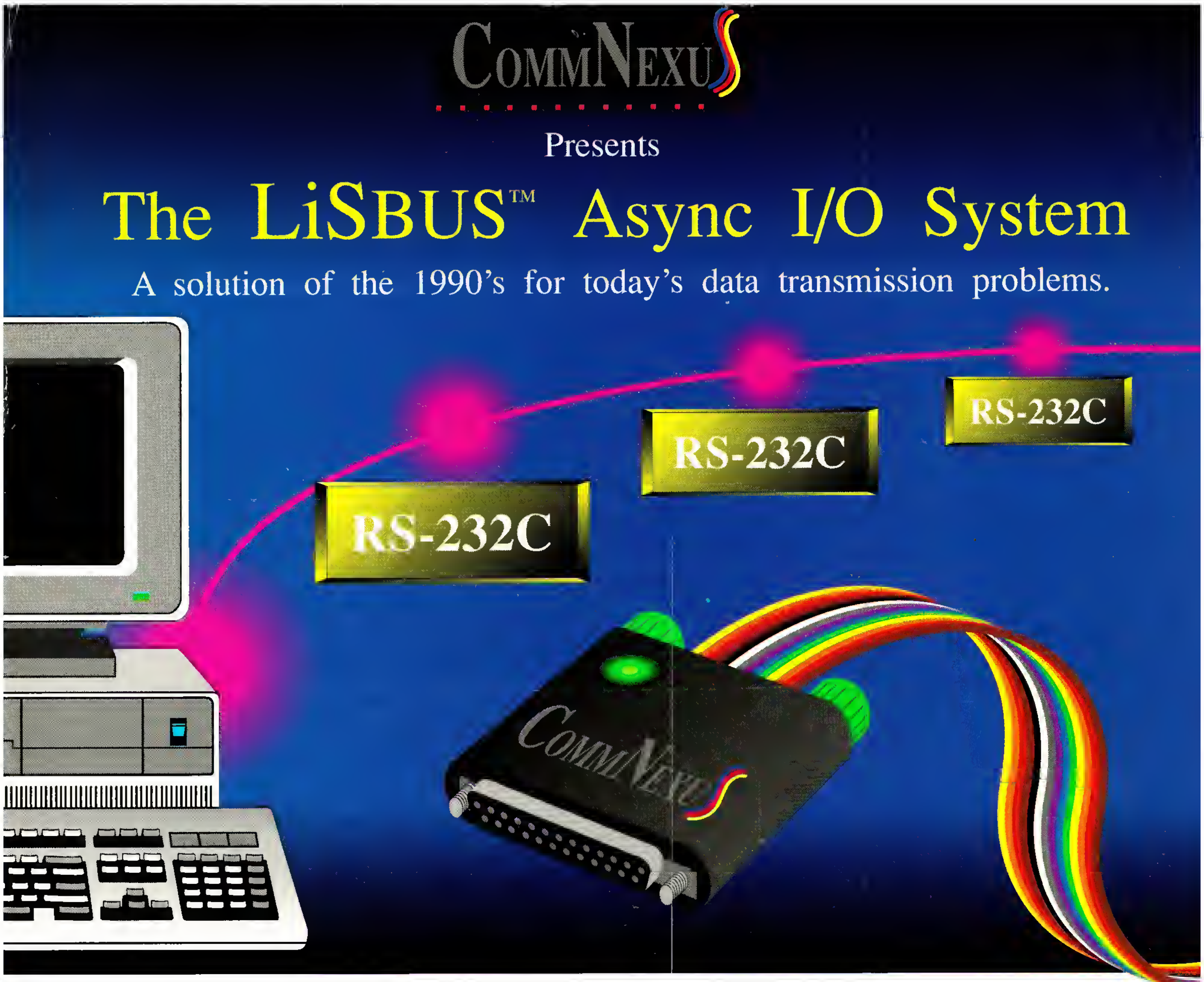
European Office
13, Ave. de L'Aquilon
B-1200 Brussels, Belgium
Phone: 32 (2) 770-21-98
Fax: 32 (2) 770-85-05

Asian Office
Ooshima Building
2-19-1 Minami-Aoyama, Minato-ku
Tokyo 107, Japan
Phone: 81 (3) 3408-3118
Fax: 81 (3) 3408-3553

IEEE OFFICERS

President: Merrill W. Buckley, Jr.
President Elect: Martha Sloan
Past President: Eric E. Sumner
Secretary: Karsten E. Drangeid
Treasurer: Theodore W. Hissey, Jr

VP, Educational Activities: Edward A. Parrish
VP, Professional Activities: Arvid G. Larson
VP, Publication Activities: James T. Cain
VP, Regional Activities: Luis T. Gandia
VP, Standards Activities: Marco W. Migliaro
VP, Technical Activities: Fernando Aldana



Outstandingly Simple and Reliable because **LiSBUS**™ is based on a breakthrough technology which uses the impedance of the bus cable to replace binary addresses. Consequently, data transmission management is greatly simplified and much more reliable than today's equivalent systems which require expensive sofware, hardware, and personnel investments.

Outstandingly Practical because is it easy to install and easy to operate. No special tools, workbench, or electronics expertise are needed. Anyone can be up and running in minutes. Just plug in the external modules and configure the system with the user-friendly **LiSBUS**™ **Link Control Software**. Each external module measures only around 2in. by 2in.

Outstandingly Flexible because a user can connect up to 60 peripherals or computers to a controlling computer through their RS-232C (COM) ports. To add peripherals, just extend the bus cable and add modules.

At an Unbeatable Price because at $650* for the **LiSBUS**™ **Starter Pack**, no alternative can offer all these advantages combined into one product without spending a much higher amount. The **Starter Pack** includes all the user needs to connect four peripherals and a complete set of **LiSBUS**™ **Development Tools** to create custom applications.

LiSBUS™:
A product of our new **CommNexus**™ line of communication systems.

Special Pre-release Offer: you can receive the **LiSBUS**™ **Starter Pack** at the special pre-release price of $550. Just call or mail in your order by February 29, 1992, at the latest. Deliveries will begin early March.

For more information and ordering contact:

In the USA and Canada:

Toll Free (800) 945-3002
Mon.-Fri. 9am - 9pm EST

GIGATEC (USA). Inc.
871 Islington Street
Portsmouth, NH 03801 USA
Tel. (603) 433-2227
Fax (603) 433-5552

In Europe:

GIGATEC SA
Ch. des Plans-Praz
1337 Vallorbe SWITZERLAND
Tel. 41 21 843 37 36
Fax 41 21 843 33 25

* specifications and prices subject to change without prior notification
Visa and MasterCard/EuroCard accepted. - CommNexus™ and LiSBUS™ are trademarks of GIGATEC SA.

Reader Service Number 1